SOLUTION OF THE DISCONTINUOUS P_1 EQUATIONS IN TWO DIMENSIONAL CARTESIAN GEOMETRY WITH TWO-LEVEL PRECONDITIONING [*]

J. S. WARSA , T. A. WAREING , AND J. E. MOREL [†]

Abstract. We present a new bilinear discontinuous (Galerkin) finite element discretization of the P_1 (spherical harmonics) equations, a first-order system of equations used for describing neutral particle radiation transport or modeling radiative transfer problems. The discrete equations are described for two-dimensional rectangular meshes; we solve the linear system with Krylov iterative methods. We have developed a novel, two-level preconditioner to improve convergence of the Krylov solvers that is based on a linear continuous finite element discretization of the diffusion equation, solved with a conjugate gradient iteration, preceded and followed by one of several different smoothing relaxations. A Fourier analysis shows that our approach is very effective over a wide range of problems. Numerical experiments confirm the results of the Fourier analysis. Computations for a realistic problem show that the preconditioner is effective and the solution method is efficient in practice.

Key words. Krylov iterative solvers, finite element methods, discontinuous Galerkin methods, multi-level preconditioning, radiative transfer, neutron transport, diffusion equation

AMS subject classifications. 65F10, 65N22, 65N30, 82D75, 85A25

1. Introduction.

In this paper we consider the solution of the discontinuous finite element (DFE) discretization of the P_1 equations. There are two potential applications in which we are interested:

1. consistent iterative convergence acceleration of particle transport calculations,
2. low order (angular) approximation to the transport equation for radiation hydrodynamics problems.

The first application involves what is well known in the radiation transport community as Diffusion Synthetic Acceleration (DSA) [2]. The convergence rate of the standard iterative method for solving the Boltzmann transport equation may be unacceptably slow when applied to thermal radiative transfer problems containing highly diffusive regions [29]. With some approximation to the transport equation, most notably the diffusion operator, convergence of the iterative solution can be dramatically accelerated by estimating the error in the scalar fluxes with the approximate operator and correcting the most recent iterate. Generally speaking, the diffusion operator must be spatially discretized in a way that is in some sense "consistent" with the discretization of the transport equation [20, 21]. This ensures that the DSA algorithm will be robust. The transport equation is often discretized with DFE methods because of their desirable numerical properties [3, 22–24, 29]. Consistency dictates that diffusion equation discretizations which are fully consistent with discontinuous transport discretizations must also be discontinuous. It is possible to define simplified discontinuous discretizations of the diffusion equation for transport acceleration purposes but they are only partially consistent, restricting their applicability [1, 28]. The DFE discretization of the P_1 equations we will describe in this paper can be viewed as discretizing the diffusion equation with a mixed, DFE method that is fully consistent with the DFE transport discretization in which we are interested [37].

[*]This work was performed under the auspices of the U.S. Department of Energy at Los Alamos National Laboratory (LANL).

[†]Los Alamos National Laboratory, P.O. Box 1663, MS D409, Los Alamos, NM 87545.

The second application is for radiation hydrodynamics problems, where equations that describe the transport of radiation are coupled with equations describing fluid flow [32]. In practice, the radiation transport is modeled with a P_1 (or diffusion) approximation to the full Boltzmann transport equation for efficiency. The DFE discretization of the P_1 equations we describe can be interpreted as an upwind Godunov method. These methods are accurate and commonly used ways to discretize fluid flow equations, such as Euler's equations [25]. A key property of Godunov discretizations is the conservation of energy, momentum, and mass over each mesh cell. Traditional discretizations of the P_1 equations generally use a staggered mesh which conserves radiation energy but does not conserve radiation momentum over a spatial mesh cell. Our method therefore locally conserves both the radiation energy and the radiation momentum, a desirable property for radiation hydrodynamics. Furthermore, there are certain advantages to using the same basic type of discretization for both the fluid dynamics and the radiation transport [26].

Our DFE discretization of the P_1 equations leads to a sparse linear system (saddle point problem) that can be written in either nonsymmetric, positive definite form or symmetric, indefinite form. This essentially new discretization is similar to some mixed, discontinuous Galerkin (dG) methods which have been applied recently to the solution of elliptic systems [4, 6, 13]. It is critical that we have an effective (in terms of improved convergence rate) and efficient (in terms of computational effort needed to achieve an adequate improvement in convergence rate) preconditioner for the Krylov iterative methods we use to solve the sparse linear system (direct methods are impractical in our applications). We have developed a novel two-level preconditioner for the DFE P_1 equations. The two level approach is based on a vertex centered, continuous finite element (CFE) diffusion equation discretization, combined with some standard pre- and post-relaxations. Information is transferred between the two levels with a sophisticated projection from the discontinuous representation to the vertex centered representation and a simple interpolation back onto the discontinuous representation. This is similar to the methods in [28], except that Morel, et al., considered a simplified discontinuous diffusion discretization that is not fully consistent with DFE transport equation discretizations. We suggest that it might be possible to use a similar two-level preconditioning approach for the iterative solution of other mixed dG discretizations.

The paper is organized as follows. First we will present the discretization of the P_1 equations in two-dimensional (x, y) geometry with rectangular meshes. Next we will discuss our preconditioning algorithm. That is followed by a Fourier analysis, confirmed by numerical experiments, that helps to identify and characterize the circumstances under which rapid convergence can be expected. The following section presents numerical results and measurements for a realistic problem to show that, in practice, our preconditioner is effective and that solutions can be computed efficiently. We conclude the paper with a few summary remarks and make recommendations for future investigation.

2. The P_1 Equations.

The P_1 equations are a system of first-order partial differential equations for the zeroth and first angular moments of the angular flux which are known as the scalar flux and current, respectively. Briefly, they can be derived by expanding the angular dependence of the flux, the quantity of interest in the Boltzmann transport equation, in a series of the spherical harmonics functions [14]. The expansion is substituted into the transport equation and angular moments of the transport equation are taken. Using the addition formula for the spherical harmonics gives a system of differential equations for the angular moments of the flux. The P_L equations are formed by truncating this system at order L to arrive at a coupled system of $L+1$ equations for the $L+1$ angular moments. Taking $L=1$ gives the P_1 equations. Thus, the P_1 equations can be looked at as an angular Galerkin approximation to the transport equation based on a spherical harmonic trial space of first order. We can select any one of several boundary conditions for the P_L equations that approximate the transport boundary conditions [31, 36]. Time and energy dependence will be neglected in this paper.

In general geometry the P_1 equations are

$$\nabla \cdot \mathbf{J}(\mathbf{r}) + \sigma_0(\mathbf{r})\Phi(\mathbf{r}) = Q_0(\mathbf{r}) \tag{2.1a}$$

$$\frac{1}{3}\nabla\Phi(\mathbf{r}) + \sigma_1(\mathbf{r})\mathbf{J}(\mathbf{r}) = \mathbf{Q}_1(\mathbf{r}), \tag{2.1b}$$

where $\Phi(\mathbf{r})$ represents the scalar flux and $\mathbf{J}(\mathbf{r})$ the current (a vector quantity) at a position $\mathbf{r}$. The cross sections $\sigma_0(\mathbf{r})$ and $\sigma_1(\mathbf{r})$ are nonnegative, usually nonzero, parameters that depend on the material in which the energetic particles of interest are moving. They are defined in terms of the angular moments of the spherical harmonic expansion of the interaction cross sections and they represent the absorption cross section, $\sigma_a(\mathbf{r})$, and the (transport corrected) total cross section $\sigma_t(\mathbf{r})$, respectively. The source terms $Q_0(\mathbf{r})$ and $\mathbf{Q}_1(\mathbf{r})$ are the zeroth and first angular moments of an inhomogeneous source. It is often assumed the source is emitting particles isotropically such that the vector $\mathbf{Q}_1(\mathbf{r})$ is zero. The first expression in the P_1 equations is called the zeroth moment, or balance, equation. The second is a vector equation whose component expression(s) are called the first moment equation(s). They will consistently be written in this order. The steady state P_1 equations are equivalent to a second order diffusion equation and later we will explain how we take advantage of this fact in solving the first-order system.

We will consider two types of boundary conditions: a type of mixed or Robin boundary condition in the the case of a vacuum boundary condition or a boundary source, or a homogeneous Neumann condition in the case of reflection on the boundary.

2.1. Two Dimensional Geometry. The P_1 equations in two-dimensional geometry on the rectangular domain $(x,y) \in [0,a] \times [0,b]$ are

$$\frac{\partial J^x}{\partial x} + \frac{\partial J^y}{\partial y} + \sigma_a(x,y)\Phi(x,y) = Q_0(x,y) \tag{2.2a}$$

$$\frac{1}{3}\frac{\partial \Phi}{\partial x} + \sigma_t(x,y)J^x(x,y) = 0 \tag{2.2b}$$

$$\frac{1}{3}\frac{\partial \Phi}{\partial y} + \sigma_t(x,y)J^y(x,y) = 0, \tag{2.2c}$$

where the current $\mathbf{J}(\mathbf{r})$ is a vector quantity whose x and y components are $J^x(x,y)$ and $J^y(x,y)$, respectively, and where we have assumed the anisotropic component of the source, $\mathbf{Q}_1(\mathbf{r})$, is zero. Using the moment equations, Eqs. 2.2b and 2.2c, to eliminate the currents in the balance equation, Eq. 2.2a, gives a diffusion equation for the scalar flux

$$-\frac{\partial}{\partial x}\left(D(x,y)\frac{\partial \Phi}{\partial x}\right) - \frac{\partial}{\partial y}\left(D(x,y)\frac{\partial \Phi}{\partial y}\right) + \sigma_a(x,y)\Phi(x,y) = Q_0(x,y), \tag{2.3}$$

where $D(x,y) = \big(3\sigma_t(x,y)\big)^{-1}$ is the diffusion coefficient.

One type of boundary conditions for the P_1 equations is of mixed type, known in the transport community as Marshak boundary conditions [14]. In the case of a vacuum boundary condition or a constant isotropic incident boundary source, the flows of inwardly and outwardly directed particles through the boundary surface $\mathbf{r}_s$ can be specified in terms of the fundamental unknowns, $\Phi(\mathbf{r}_s)$ and $\mathbf{J}(\mathbf{r}_s)$. The Marshak boundary conditions express a fixed relationship between the two quantities. They are an approximation to the transport boundary conditions consistent with the P_1 approximation of the transport equation. Assuming a source or vacuum condition along the entire length of any of the four sides of the problem, the Marshak conditions are

$$\frac{1}{4}\Phi(0,y)+\frac{1}{2}J^x(0,y)=J_L^+,\quad \frac{1}{4}\Phi(a,y)-\frac{1}{2}J^x(a,y)=J_R^-,\quad y\in[0,b], \tag{2.4a}$$

$$\frac{1}{4}\Phi(x,0)+\frac{1}{2}J^x(x,0)=J_B^+,\quad \frac{1}{4}\Phi(x,b)-\frac{1}{2}J^x(x,b)=J_T^-,\quad x\in[0,a], \tag{2.4b}$$

where the values J_L^+, J_R^-, J_B^+ and J_T^- represent the (isotropic) source of particles entering a problem through the left, right, bottom and top boundary surfaces, respectively. They are zero in the case of a vacuum.

Reflective (homogeneous Neumann) boundary conditions physically represent a zero particle flow condition, that is,

$$J^x(0,y)=0,\quad J^x(a,y)=0,\quad y\in[0,b], \tag{2.5a}$$

$$J^y(x,0)=0,\quad J^y(x,b)=0,\quad x\in[0,a]. \tag{2.5b}$$

Source or vacuum boundary conditions for the diffusion equation are obtained by using Eqs. 2.2b and 2.2c to eliminate the currents in Eqs. 2.4:

$$\frac{1}{4}\Phi(0,y)-\frac{1}{2}D(0,y)\left.\frac{\partial\Phi}{\partial x}\right|_{x=0}=J_L^+,\quad \frac{1}{4}\Phi(a,y)+\frac{1}{2}D(a,y)\left.\frac{\partial\Phi}{\partial x}\right|_{x=a}=J_R^-,\quad y\in[0,b], \tag{2.6a}$$

$$\frac{1}{4}\Phi(x,0)-\frac{1}{2}D(x,0)\left.\frac{\partial\Phi}{\partial y}\right|_{y=0}=J_B^+,\quad \frac{1}{4}\Phi(x,b)+\frac{1}{2}D(x,b)\left.\frac{\partial\Phi}{\partial y}\right|_{y=b}=J_T^-,\quad x\in[0,a]. \tag{2.6b}$$

Reflection conditions are simply

$$\left.\frac{\partial\Phi}{\partial x}\right|_{x=0}=0,\quad \left.\frac{\partial\Phi}{\partial x}\right|_{x=a}=0,\quad y\in[0,b], \tag{2.7a}$$

$$\left.\frac{\partial\Phi}{\partial y}\right|_{y=0}=0,\quad \left.\frac{\partial\Phi}{\partial y}\right|_{y=b}=0,\quad x\in[0,a]. \tag{2.7b}$$

3. Discretization of the P_1 Equations.

We will now describe the DFE discretization of P_1 system of equations. We use a Galerkin finite element method with bilinear basis functions and assume piecewise constant cross sections and distributed sources (material properties). Discontinuities are introduced using physical principles. The coefficient matrix is "lumped" by approximating integrations with the trapezoidal rule. We employ a standard (SPD) CFE discretization of the diffusion equation that forms the basis of our two-level preconditioner. We will describe how to derive this discretization directly from the DFE P_1 equations.

3.1. Bilinear Discontinuous Galerkin Finite Elements. Dividing the domain $[0,a]\times[0,b]$ into discrete points $x_i, i=0,N_x$, and $y_j, j=0,N_y$, defines a two-dimensional grid of N_x spatial mesh cells of widths $h_i, i=1,N_x$, in the x–dimension and N_y spatial mesh cells of widths $k_j, j=1,N_y$, in the y–dimension. Piecewise constant cross sections $\sigma_{t,i,j}, \sigma_{a,i,j}$, are given for $i=1,N_x, j=1,N_y$.

Reference to Fig. 1 should help in visualizing the two-dimensional DFE discretization that we present now. On a mesh cell (i,j), the following bilinear approximations are made for the scalar flux and the x and y current components:

$$\begin{aligned}\Phi_{i,j}(x,y) &= \Phi_{LB,i,j}B_{LB}(x,y)+\Phi_{RB,i,j}B_{RB}(x,y)\\ &\quad+\Phi_{LT,i,j}B_{LT}(x,y)+\Phi_{RT,i,j}B_{RT}(x,y) \qquad (3.1a)\\ J^x_{i,j}(x,y) &= J^x_{LB,i,j}B_{LB}(x,y)+J^x_{RB,i,j}B_{RB}(x,y)\\ &\quad+J^x_{LT,i,j}B_{LT}(x,y)+J^x_{RT,i,j}B_{RT}(x,y) \qquad (3.1b)\\ J^y_{i,j}(x,y) &= J^y_{LB,i,j}B_{LB}(x,y)+J^y_{RB,i,j}B_{RB}(x,y)\\ &\quad+J^y_{LT,i,j}B_{LT}(x,y)+J^y_{RT,i,j}B_{RT}(x,y). \qquad (3.1c)\end{aligned}$$

The bilinear basis functions are defined by compositions of one dimensional linear Lagrange functions on a cell. They have the property that they are unity in their respective corners ("L" meaning left, "R" meaning right, "B" meaning bottom, and "T" meaning top) and zero in the other three corners. Therefore, the functions in Eqs. 3.1 take on the value of the corresponding expansion coefficient when evaluated at a corner of the cell.

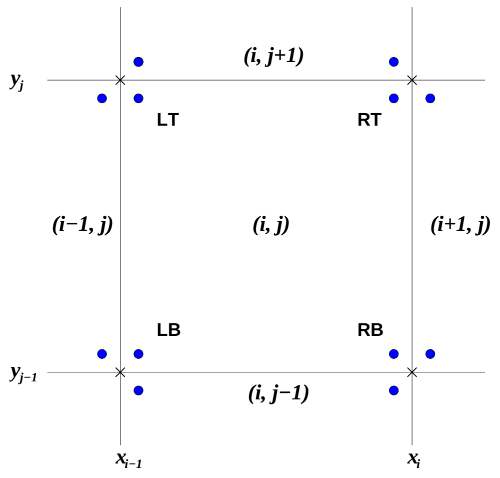


Fig. 1. Mesh cell (i,j) illustrating coupling to neighboring cells. Locations of the relevant DFE unknowns are indicated by $\bullet$ symbols surrounding the vertices. The scalar fluxes of the CFE discretization reside on the vertices, indicated by the $\times$ symbols.

In the standard Galerkin procedure, we substitute Eqs. 3.1 into Eqs. 2.2 and take inner products with respect to the basis functions to get the following weak form for the discrete solution:

$$\begin{aligned}
&\Big(B_f(x_i,y),\, J_b^x(x_i,y)\Big)_y - \Big(B_f(x_{i-1},y),\, J_b^x(x_{i-1},y)\Big)_y - \Big(\frac{\partial B_f}{\partial x},\, J_{i,j}^x\Big)_{xy} \\
&+\Big(B_f(x,y_j),\, J_b^y(x,y_j)\Big)_x - \Big(B_f(x,y_{j-1}),\, J_b^y(x,y_{j-1})\Big)_x - \Big(\frac{\partial B_f}{\partial y},\, J_{i,j}^y\Big)_{xy} \qquad (3.2a)\\
&\quad + \sigma_{a,i,j}\Big(B_f,\, \Phi_{i,j}\Big)_{xy} = \Big(B_f,\, Q_0\Big)_{xy},
\end{aligned}$$

$$\begin{aligned}
&\frac{1}{3}\bigg(\Big(B_f(x_i,y),\, \Phi_b(x_i,y)\Big)_y - \Big(B_f(x_{i-1},y),\, \Phi_b(x_{i-1},y)\Big)_y - \Big(\frac{\partial B_f}{\partial x},\, \Phi_{i,j}\Big)_{xy}\bigg) \\
&\quad + \sigma_{t,i,j}\Big(B_f,\, J_{i,j}^x\Big)_{xy} = 0,
\end{aligned} \qquad (3.2b)$$

$$\begin{aligned}
&\frac{1}{3}\bigg(\Big(B_f(x,y_j),\, \Phi_b(x,y_j)\Big)_x - \Big(B_f(x,y_{j-1}),\, \Phi_b(x,y_{j-1})\Big)_x - \Big(\frac{\partial B_f}{\partial y},\, \Phi_{i,j}\Big)_{xy}\bigg) \\
&\quad + \sigma_{t,i,j}\Big(B_f,\, J_{i,j}^y\Big)_{xy} = 0,
\end{aligned} \qquad (3.2c)$$

where any of the four basis functions in Eqs. 3.1 can be represented by $B_f(x,y)$ for $f = LB, RB, LT$, or RT. The inner products above are computed for each cell (i,j) over $x \in [x_{i-1}, x_i]$ and and $y \in [y_{j-1}, y_j]$, that is,

$$\Big(u,\, v\Big)_x = \int_{x_{i-1}}^{x_i} u(x)\, v(x)\, dx, \qquad (3.3a)$$

$$\Big(u,\, v\Big)_y = \int_{y_{j-1}}^{y_j} u(y)\, v(y)\, dy, \qquad (3.3b)$$

$$\Big(u,\, v\Big)_{xy} = \int_{x_{i-1}}^{x_i}\int_{y_{j-1}}^{y_j} u(x,y)\, v(x,y)\, dy\, dx. \qquad (3.3c)$$

We can lump the equations by evaluating these integrals approximately, using with the trapezoidal rule in each dimension:

$$\Big(u,\, v\Big)_x = \frac{\Delta x_i}{2}\Big[\, u(x_{i-1})\, v(x_{i-1}) + u(x_i)\, v(x_i)\,\Big] \qquad (3.4a)$$

$$\Big(u,\, v\Big)_y = \frac{\Delta y_j}{2}\Big[\, u(y_{j-1})\, v(y_{j-1}) + u(y_j)\, v(y_j)\,\Big] \qquad (3.4b)$$

$$\begin{aligned}
\Big(u,\, v\Big)_{xy} = \frac{\Delta x_i \Delta y_j}{2}\Big[\, & u(x_{i-1},y_{j-1})\, v(x_{i-1},y_{j-1}) + u(x_i,y_{j-1})\, v(x_i,y_{j-1}) \\
& + u(x_{i-1},y_j)\, v(x_{i-1},y_j) + u(x_i,y_j)\, v(x_i,y_j)\,\Big].
\end{aligned} \qquad (3.4c)$$

3.2. DFE Discretization of the $\mathbf{P}_1$ Equations. We used Green's Theorem in writing the weak form given in Eqs. 3.2, giving rise to the "boundary" terms in those expressions, denoted by the subscript b. Because basis functions from adjacent cells overlap along the cell edges these terms are not uniquely defined. A unique definition can generally be obtained by "upwinding". Upwinding not only defines the cell edge values but also produces a discretization that is discontinuous. Upwinding is implemented naturally when flow directions can be associated with the variables of interest. In that case, the boundary terms are defined in terms of the flows into and out of a cell through the cell edges. However, while there is no particular flow direction associated with the scalar flux or currents in the P$_1$ equations, there are several ways we can relate the scalar flux and current to incoming and outgoing flows through a cell edge. For instance, we could diagonalize the (hyperbolic) P$_1$ equations with an eigenvector decomposition, as is done in upwinded Godunov methods, reformulating the problem in terms of so called characteristic variables, to which can be associated well defined direction of flow [18, 25].

Alternatively, we can use the physical interpretation of the P$_1$ approximation, which assumes the angular dependence of the full particle distribution is at most linear in angle [7]. Partial currents are defined as a weighted projection of the particle distribution flowing in through a surface or out through a surface, relative to the surface normal. The partial currents can be used to decompose the particle flow through a surface into two directions, the flow of inwardly directed particles and the flow of outwardly directed particles. This is the same as the Marshak boundary conditions, except that the partial currents are used to relate $\Phi(x,y)$ and $\mathbf{J}(x,y)$ to the particle flows through a cell edge instead of through the boundary.

We will describe this in detail for rectangular cells. First we write down the partial currents at a surface at $x = \hat{x}$, in the positive and negative x–direction, respectively, as

$$J^{x+}(\hat{x},y) = \frac{1}{4}\Phi(\hat{x},y) + \frac{1}{2}J^{x}(\hat{x},y) \tag{3.5a}$$

$$J^{x-}(\hat{x},y) = \frac{1}{4}\Phi(\hat{x},y) - \frac{1}{2}J^{x}(\hat{x},y), \tag{3.5b}$$

We can then write the "boundary" terms for the scalar flux and current at $x = \hat{x}$ as

$$\Phi_b(\hat{x},y) = 2\left[J^{x+}(\hat{x},y) + J^{x-}(\hat{x},y)\right] \tag{3.6a}$$

$$J^{x}_{b}(\hat{x},y) = J^{x+}(\hat{x},y) - J^{x-}(\hat{x},y). \tag{3.6b}$$

Now, the partial currents at a surface at $y = \hat{y}$ in the positive and negative y–directions are, respectively,

$$J^{y+}(x,\hat{y}) = \frac{1}{4}\Phi(x,\hat{y}) + \frac{1}{2}J^{y}(x,\hat{y}) \tag{3.7a}$$

$$J^{y-}(x,\hat{y}) = \frac{1}{4}\Phi(x,\hat{y}) - \frac{1}{2}J^{y}(x,\hat{y}). \tag{3.7b}$$

Similarly, we can then write the boundary terms for the scalar flux and current at $y = \hat{y}$ as

$$\Phi_b(x,\hat{y}) = 2\left[J^{y+}(x,\hat{y}) + J^{y-}(x,\hat{y})\right] \tag{3.8a}$$

$$J^{y}_{b}(x,\hat{y}) = J^{y+}(x,\hat{y}) - J^{y-}(x,\hat{y}). \tag{3.8b}$$

Because Eqs. 3.5 and 3.7 and the Marshak boundary conditions, Eqs. 2.4, come from the same approximation, we can easily incorporate boundary conditions into our discretization. First, we define the factors ξ_k as follows. Letting the subscript f take on the values L for the left boundary ($x = 0$), R for the right ($x = a$), B for the bottom ($y = 0$), and T for the top ($y = b$), we set ξ_k according to

$$\xi_k = \begin{cases} 1, & \text{for reflective boundary conditions} \\ 0, & \text{for vacuum boundary condition} \\ & \text{or if the cell edge is not on the domain boundary} \end{cases}$$

These factors, together with Eq. 3.5 through Eq. 3.8, are used to express the flow of particles through the surfaces of a cell (i,j). The flow for particles emerging from a cell (i,j) in the negative x–direction through $\hat{x}=x_{i-1}$ and in the positive x–direction through $\hat{x}=x_i$ are, respectively, for $y\in[y_{j-1},y_j]$

$$J^{x-}(x_{i-1},y)=\frac{1}{4}\Phi_{i,j}(x_{i-1},y)-\frac{1}{2}J^x_{i,j}(x_{i-1},y) \tag{3.9a}$$

$$J^{x+}(x_i,y)=\frac{1}{4}\Phi_{i,j}(x_i,y)+\frac{1}{2}J^x_{i,j}(x_i,y). \tag{3.9b}$$

The flow for particles entering cell (i,j) in the positive x–direction through $\hat{x}=x_{i-1}$ and in the negative x–direction $\hat{x}=x_i$ are given in terms of the bilinear expansions for the quantities in the adjacent cells on the left and right, respectively,

$$J^{x+}(x_{i-1},y)=(1-\xi_L)\left(\frac{1}{4}\Phi_{i-1,j}(x_{i-1},y)-\frac{1}{2}J^x_{i-1,j}(x_{i-1},y)\right)+\xi_L J^{x-}(x_{i-1},y) \tag{3.10a}$$

$$J^{x-}(x_i,y)=(1-\xi_R)\left(\frac{1}{4}\Phi_{i+1,j}(x_i,y)+\frac{1}{2}J^x_{i+1,j}(x_i,y)\right)+\xi_R J^{x+}(x_i,y). \tag{3.10b}$$

The expressions in Eqs. 3.9 and 3.10 are substituted into Eqs. 3.6 for $\hat{x}=x_{i-1}$ and then for $\hat{x}=x_i$ which in turn define the cell edge values $\Phi_b(x_{i-1},y)$, $\Phi_b(x_i,y)$ and $J^x_b(x_{i-1},y)$, $J^x_b(x_i,y)$ in Eqs. 3.2. The flow for particles emerging from cell (i,j) in the negative y–direction through $\hat{y}=y_{j-1}$ and in the positive y–direction $\hat{y}=y_j$ in terms bilinear expansions for the quantities are, respectively, for $x\in[x_{i-1},x_i]$

$$J^{y-}(x,y_{j-1})=\frac{1}{4}\Phi_{i,j}(x,y_{j-1})-\frac{1}{2}J^y_{i,j}(x,y_{j-1}) \tag{3.11a}$$

$$J^{y+}(x,y_j)=\frac{1}{4}\Phi_{i,j}(x,y_j)+\frac{1}{2}J^y_{i,j}(x,y_j). \tag{3.11b}$$

The flow for particles entering cell (i,j) in the positive y–direction through $\hat{y}=y_{j-1}$ and in the negative y–direction $\hat{y}=y_j$ are given in terms of the bilinear expansions for the quantities in the adjacent cells above and below, respectively,

$$J^{y+}(x,y_{j-1})=(1-\xi_B)\left(\frac{1}{4}\Phi_{i,j-1}(x,y_{j-1})-\frac{1}{2}J^y_{i,j-1}(x,y_{j-1})\right)+\xi_B J^{y-}(x,y_{j-1}) \tag{3.12a}$$

$$J^{y-}(x,y_j)=(1-\xi_T)\left(\frac{1}{4}\Phi_{i+1,j}(x,y_j)+\frac{1}{2}J^x_{i+1,j}(x,y_j)\right)+\xi_T J^{y+}(x,y_j). \tag{3.12b}$$

The expressions in Eqs. 3.11 and 3.12 are substituted into Eqs. 3.8 for $\hat{y}=y_{j-1}$ and $\hat{y}=y_j$ which in turn define the cell edge values $\Phi_b(x,y_{j-1})$, $\Phi_b(x,y_j)$, and $J^y_b(x,y_{j-1})$, $J^y_b(x,y_j)$, in Eqs. 3.2.

Using these definitions of the cell edge values, together with the bilinear expansions Eqs. 3.1, a discontinuous discretization is derived for each cell consisting of twelve equations (four equations for the balance equation and four for each of the two first moment equations) in twelve unknowns (four for the scalar flux, and four for each of the two current components). For a cell (i,j), the solution vector is ordered as

$$\bar{x}_{i,j}=[\Phi_{LB},\ \Phi_{RB},\ \Phi_{LT},\ \Phi_{RT},J^x_{LB},\ J^x_{RB},\ J^x_{LT},\ J^x_{RT},J^y_{LB},\ J^y_{RB},\ J^y_{LT},\ J^y_{RT}]^T.$$

We order the grid in rows, left-to-right, starting from the bottom so that the solution vector is stored as

$$\bar{x} = [\bar{x}_{1,1},\ \bar{x}_{2,1},\ \dots,\ \bar{x}_{N_x,1}, \bar{x}_{1,2},\ \bar{x}_{2,2},\ \dots,\ \bar{x}_{N_x,2},\ \dots, \bar{x}_{1,N_y},\ \bar{x}_{2,N_y},\ \dots,\ \bar{x}_{N_x,N_y}]^T. \tag{3.13}$$

The two-dimensional lumped, bilinear DFE discretization of the P$_1$ equations can be written in matrix form as follows.

$$\mathbf{A}_{i,j}^{1,1} = \begin{bmatrix} \alpha_{L,B} & 0 & 0 & 0 \\ 0 & \alpha_{R,B} & 0 & 0 \\ 0 & 0 & \alpha_{L,T} & 0 \\ 0 & 0 & 0 & \alpha_{R,T} \end{bmatrix}$$

$$\mathbf{A}_{i}^{2,2} = \begin{bmatrix} \beta_L & 0 & 0 & 0 \\ 0 & \beta_R & 0 & 0 \\ 0 & 0 & \beta_L & 0 \\ 0 & 0 & 0 & \beta_R \end{bmatrix} \quad \mathbf{A}_{j}^{3,3} = \begin{bmatrix} \gamma_B & 0 & 0 & 0 \\ 0 & \gamma_B & 0 & 0 \\ 0 & 0 & \gamma_T & 0 \\ 0 & 0 & 0 & \gamma_T \end{bmatrix}$$

$$\mathbf{A}_{j}^{1,2} = \begin{bmatrix} \xi_L k_j & k_j & 0 & 0 \\ -k_j & -\xi_R k_j & 0 & 0 \\ 0 & 0 & \xi_L k_j & k_j \\ 0 & 0 & -k_j & -\xi_R k_j \end{bmatrix} \quad \mathbf{A}_{j}^{2,1} = \begin{bmatrix} -\xi_L k_j & k_j & 0 & 0 \\ -k_j & \xi_R k_j & 0 & 0 \\ 0 & 0 & -\xi_L k_j & k_j \\ 0 & 0 & -k_j & \xi_R k_j \end{bmatrix}$$

$$\mathbf{A}_{i}^{1,3} = \begin{bmatrix} \xi_B h_i & 0 & h_i & 0 \\ 0 & \xi_B h_i & 0 & h_i \\ -h_i & 0 & -\xi_T h_i & 0 \\ 0 & -h_i & 0 & -\xi_T h_i \end{bmatrix} \quad \mathbf{A}_{i}^{3,1} = \begin{bmatrix} -\xi_B h_i & 0 & h_i & 0 \\ 0 & -\xi_B h_i & 0 & h_i \\ -h_i & 0 & \xi_T h_i & 0 \\ 0 & -h_i & 0 & \xi_T h_i \end{bmatrix}$$

$$\mathbf{A}_{i,j} = \begin{bmatrix} \frac{1}{8}\mathbf{A}_{i,j}^{1,1} & \frac{1}{4}\mathbf{A}_{j}^{1,2} & \frac{1}{4}\mathbf{A}_{i}^{1,3} \\ \frac{1}{6}\mathbf{A}_{j}^{2,1} & \frac{1}{6}\mathbf{A}_{i}^{2,2} & 0 \\ \frac{1}{6}\mathbf{A}_{i}^{3,1} & 0 & \frac{1}{6}\mathbf{A}_{j}^{3,3} \end{bmatrix} \tag{3.14}$$

$$\mathbf{S}_i = \begin{bmatrix} 0 & 0 & -h_i & 0 \\ 0 & 0 & 0 & -h_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{R}_j = \begin{bmatrix} 0 & -k_j & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -k_j \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{W}_j = \begin{bmatrix} \frac{1}{8}\mathbf{R}_j & \frac{1}{4}\mathbf{R}_j & 0 \\ \frac{1}{12}\mathbf{R}_j & \frac{1}{6}\mathbf{R}_j & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{X}_j = \begin{bmatrix} \frac{1}{8}\mathbf{R}_j^T & -\frac{1}{4}\mathbf{R}_j^T & 0 \\ -\frac{1}{12}\mathbf{R}_j^T & \frac{1}{6}\mathbf{R}_j^T & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.15}$$

$$\mathbf{Y}_i = \begin{bmatrix} \frac{1}{8}\mathbf{S}_i & 0 & \frac{1}{4}\mathbf{S}_i \\ 0 & 0 & 0 \\ \frac{1}{12}\mathbf{S}_i & 0 & \frac{1}{6}\mathbf{S}_i \end{bmatrix} \qquad \mathbf{Z}_i = \begin{bmatrix} \frac{1}{8}\mathbf{S}_i^T & 0 & -\frac{1}{4}\mathbf{S}_i^T \\ 0 & 0 & 0 \\ -\frac{1}{12}\mathbf{S}_i^T & 0 & \frac{1}{6}\mathbf{S}_i^T \end{bmatrix} \tag{3.16}$$

The auxiliary quantities appearing in the matrices are defined as

$$\alpha_{f,g} = 2\sigma_{a,i,j} h_i k_j + (1-\xi_f)k_j + (1-\xi_g)h_i \tag{3.17a}$$
$$\beta_f = 3\sigma_{t,i,j} h_i + 2(1+\xi_f) \tag{3.17b}$$
$$\gamma_f = 3\sigma_{t,i,j} k_j + 2(1+\xi_f). \tag{3.17c}$$

Following the global cell ordering given in Eq. 3.13, the matrix is filled in 12×12 blocks according to

$$\mathbf{A}_{i,j}\bar{x}_{i,j} + \mathbf{W}_j\bar{x}_{i-1,j} + \mathbf{X}_j\bar{x}_{i+1,j} + \mathbf{Y}_i\bar{x}_{i,j-1} + \mathbf{Z}_i\bar{x}_{i,j+1} = \bar{b}_{i,j}, \tag{3.18}$$

the cells being coupled in a block, five-point stencil as shown in Fig. 1.

We allow the isotropic source to be specified by piecewise constants $Q_{0,i,j}, i = 1, N_x, j = 1, N_y$, on each cell. Then the bilinear source representation gives the following source vector, $\bar{b}_{i,j}$, for a cell (i,j):

$$\bar{b}_{i,j} = \frac{1}{4}h_i k_j [Q_{0,i,j},\ Q_{0,i,j},\ Q_{0,i,j},\ Q_{0,i,j},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]^T. \tag{3.19}$$

Particles can also enter the system through the boundaries. We consider only isotropically emitted sources of particles. We compute the values J_L^+, J_R^-, J_B^+ and J_T^-, which is done by evaluating the incoming partial currents that correspond to isotropic boundary sources. The incident partial currents are then used in the boundary conditions, Eqs. 2.4. They are assumed constant along any of the faces for which such boundary conditions are specified. If i or j lies on the boundary of the problem, the quantities in Eq. 3.18 corresponding to nonexistent cells are replaced by the source terms J_L^+, J_R^-, J_B^+ and J_T^- and moved to the right-hand side of Eq. 3.18. Thus source boundary conditions enter naturally into the discrete equations and the source vector is modified as follows. If a source is present on the left or right face,

$$\bar{b}_{1,j} = \bar{b}_{1,j} + J_L^+ k_j [\frac{1}{2},\ \frac{1}{2},\ 0,\ 0,\ \frac{1}{3},\ \frac{1}{3},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]^T \tag{3.20a}$$

$$\bar{b}_{N_x,j} = \bar{b}_{N_x,j} + J_R^- k_j [0,\ 0,\ \frac{1}{2},\ \frac{1}{2},\ 0,\ 0,\ -\frac{1}{3},\ -\frac{1}{3},\ 0,\ 0,\ 0,\ 0]^T, \tag{3.20b}$$

respectively, for all $j = 1, N_y$. If a source is present on the bottom or top face,

$$\bar{b}_{i,1} = \bar{b}_{i,1} + J_B^+ h_i [\frac{1}{2},\ 0,\ \frac{1}{2},\ 0,\ 0,\ 0,\ 0,\ 0,\ \frac{1}{3},\ 0,\ \frac{1}{3},\ 0]^T \tag{3.21a}$$

$$\bar{b}_{i,N_y} = \bar{b}_{i,N_y} + J_T^- h_i [0,\ \frac{1}{2},\ 0,\ \frac{1}{2},\ 0,\ 0,\ 0,\ 0,\ 0,\ -\frac{1}{3},\ 0,\ -\frac{1}{3}]^T, \tag{3.21b}$$

respectively, for all $i = 1, N_x$.

In this paper, we chose a cell-wise ordering over the problem domain followed by the unknowns Φ at the vertices and then $\mathbf{J}$ at the vertices. If we instead ordered the problems first by Φ over all cells and then vertices, followed by the vector $\mathbf{J}$ over all cells and then vertices, we could write the linear system in the nonsymmetric 2×2 block form

$$\begin{bmatrix} \mathbf{A}_a & -\mathbf{A}_0^T \\ \frac{1}{3}\mathbf{A}_0 & \mathbf{A}_t \end{bmatrix} \begin{bmatrix} \Phi \\ \mathbf{J} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ g \end{bmatrix}. \tag{3.22}$$

We omit the details of the blocks for brevity, although it is possible to deduce their meaning by comparison with Eqs. 2.1 or 2.2. We simply wish to point out this more traditional saddle point formulation that arises from mixed finite element methods for elliptic operators. Such problems have been well studied; see [10, 12, 15, 33, 34], for example. We have observed that the submatrices $\mathbf{A}_t$ and $\mathbf{A}_a$ are symmetric positive definite (SPD) so that the full linear system is positive definite. The linear system can then also be written in the symmetric form

$$\begin{bmatrix} -\mathbf{A}_a & \mathbf{A}_0^T \\ \mathbf{A}_0 & 3\mathbf{A}_t \end{bmatrix} \begin{bmatrix} \Phi \\ \mathbf{J} \end{bmatrix} = \begin{bmatrix} -g \\ \mathbf{0} \end{bmatrix}. \tag{3.23}$$

In this case the linear system is indefinite which we can solve with the iterative methods SYMMLQ or MINRES. Both methods are computationally efficient and guaranteed to converge but they require SPD preconditioners. Unfortunately we will see that our preconditioner is nonsymmetric.

3.3. CFE Discretization of the Diffusion Equation. The CFE discretization of the diffusion equation is derived by combining the DFE P_1 equations to obtain a discrete equation for the scalar flux. We will illustrate the procedure by manipulating the first of Eqs. 3.2 for $f = LB$ with the right-hand side set to an arbitrary term corresponding to the discontinuous scalar flux:

$$\begin{aligned} \frac{1}{4}\sigma_{a,i,j} h_i k_j \Phi_{LB,i,j} + \frac{1}{8} k_j \big(\Phi_{LB,i,j} - \Phi_{RB,i-1,j}\big) + \frac{1}{8} h_i \big(\Phi_{LB,i,j} - \Phi_{LT,i,j-1}\big) \\ + \frac{1}{4} k_j \big(J^x_{RB,i,j} - J^x_{RB,i-1,j}\big) + \frac{1}{4} h_i \big(J^y_{LT,i,j} - J^y_{LT,i,j-1}\big) = r^0_{LB,i,j}. \end{aligned} \tag{3.24}$$

The rest of Eqs. 3.2 will also have their right-hand sides set to discontinuous unknowns. In practice, the right-hand side is an unspecified vector with the same ordering as the discontinuous unknowns.

The first step is to allow the discontinuous quantities from the four cells surrounding a grid vertex to take on a single value. Recall that the four cell vertices of a cell (i, j) are located at (x_{i-1}, y_{j-1}), (x_i, y_{j-1}), (x_{i-1}, y_j), and (x_i, y_j) corresponding to the lower-left, lower-right, upper-left and upper-right corners, respectively. Then, for example, the following assignments are made for the lower-left cell vertex at $i-1, j-1$:

$$\Phi_{LB,i,j}, \Phi_{RB,i-1,j}, \Phi_{RT,i-1,j-1}, \Phi_{LT,i,j-1} \longrightarrow \varphi_{i-1,j-1}, \tag{3.25a}$$

$$J^x_{LB,i,j}, J^x_{RB,i-1,j}, J^x_{RT,i-1,j-1}, J^x_{LT,i,j-1} \longrightarrow j^x_{i-1,j-1}, \tag{3.25b}$$

$$J^y_{LB,i,j}, J^y_{RB,i-1,j}, J^y_{RT,i-1,j-1}, J^y_{LT,i,j-1} \longrightarrow j^y_{i-1,j-1}. \tag{3.25c}$$

These and similar assignments are made for the four vertices on a cell (i,j) in all twelve of the discontinuous P_1 equations in Eqs. 3.2. Our example, Eq. 3.24, becomes

$$\frac{1}{4}\sigma_{a,i,j}h_i k_j \varphi_{i-1,j-1} + \frac{1}{4}k_j\big(j^x_{i,j-1} - j^x_{i-1,j-1}\big) + \frac{1}{4}h_i\big(j^y_{i-1,j} - j^y_{i-1,j-1}\big) = r^0_{LB,i,j} \tag{3.26}$$

The second step involves the four balance equations only. After having substituted for the continuous quantities, the balance equations are "shifted" to the vertex i,j. That is, the balance equation for $f = RT$ is unchanged, while the balance equation for $f = RB$ is written for $j = j+1$ and that for $f = LT$ is written for $i = i+1$. Our sample equation for $f = LB$ is written for $i = i+1, j = j+1$ and Eq. 3.26 becomes

$$\frac{1}{4}\sigma_{a,i+1,j+1}h_{i+1}k_{j+1}\varphi_{i,j} + \frac{1}{4}k_{j+1}\big(j^x_{i+1,j} - j^x_{i,j}\big) + \frac{1}{4}h_{i+1}\big(j^y_{i,j+1} - j^y_{i,j}\big) = r^0_{LB,i+1,j+1}. \tag{3.27}$$

The third step is to manipulate the moment equations, with the continuous values already substituted, to find expressions for the continuous currents in terms of the continuous scalar fluxes in order to eliminate the currents in the four balance equations. For our example, we first add the x–component current equations for $f = LT$ and $f = RT$,

$$\frac{1}{3}\big(\varphi_{i,j-1} - \varphi_{i-1,j-1}\big) + \frac{1}{2}\sigma_{tr,i,j}h_i\big(j^x_{i,j-1} + j^x_{i-1,j-1}\big) = r^x_{LB,i,j} + r^x_{RB,i,j}, \tag{3.28a}$$

and the y–component current equations for $f = RB$ and $f = RT$,

$$\frac{1}{3}\big(\varphi_{i-1,j} - \varphi_{i-1,j-1}\big) + \frac{1}{2}\sigma_{tr,i,j}k_j\big(j^y_{i-1,j} + j^y_{i-1,j-1}\big) = r^y_{LB,i,j} + r^y_{LT,i,j}. \tag{3.28b}$$

The previous two expressions are solved for $j^x_{i,j+1}$ and $j^y_{i+1,j}$, respectively,

$$j^x_{i,j-1} = -j^x_{i-1,j-1} + \frac{2}{(3\sigma_{tr,i,j}h_i)}\big(\varphi_{i-1,j-1} - \varphi_{i,j-1}\big) + \frac{2}{(\sigma_{tr,i,j}h_i)}\big(r^x_{LB,i,j} + r^x_{RB,i,j}\big) \tag{3.29a}$$

$$j^y_{i-1,j} = -j^y_{i-1,j-1} + \frac{2}{(3\sigma_{tr,i,j}k_j)}\big(\varphi_{i-1,j-1} - \varphi_{i-1,j}\big) + \frac{2}{(\sigma_{tr,i,j}k_j)}\big(r^y_{LB,i,j} + r^y_{LT,i,j}\big), \tag{3.29b}$$

which are both shifted by setting $i = i+1, j = j+1$:

$$\begin{aligned} j^x_{i+1,j} = -j^x_{i,j} &+ \frac{2}{(3\sigma_{tr,i+1,j+1}h_{i+1})}\big(\varphi_{i+1,j} - \varphi_{i,j}\big) \\ &+ \frac{2}{(\sigma_{tr,i+1,j+1}h_{i+1})}\big(r^x_{LB,i+1,j+1} + r^x_{RB,i+1,j+1}\big) \end{aligned} \tag{3.30a}$$

$$\begin{aligned} j^y_{i,j+1} = -j^y_{i,j} &+ \frac{2}{(3\sigma_{tr,i+1,j+1}k_{j+1})}\big(\varphi_{i,j} - \varphi_{i,j+1}\big) \\ &+ \frac{2}{(\sigma_{tr,i+1,j+1}k_{j+1})}\big(r^y_{LB,i+1,j+1} + r^y_{LT,i+1,j+1}\big). \end{aligned} \tag{3.30b}$$

These expressions are then substituted into Eq. 3.27 to obtain

$$\begin{aligned}
&\frac{1}{4}\sigma_{a,i+1,j+1}h_{i+1}k_{j+1}\varphi_{i,j} - \frac{1}{2}k_{j+1}j^x_{i,j} - \frac{1}{2}h_{i+1}j^y_{i,j} \\
&\quad + \frac{h_{i+1}}{(6\sigma_{tr,i+1,j+1}k_{j+1})}\big(\varphi_{i,j} - \varphi_{i,j+1}\big) + \frac{k_{j+1}}{(6\sigma_{tr,i+1,j+1}h_{i+1})}\big(\varphi_{i,j} - \varphi_{i+1,j}\big) \\
&\quad = r^0_{LB,i+1,j+1} \qquad (3.31)\\
&\quad - \frac{k_{j+1}}{(2\sigma_{tr,i+1,j+1}h_{i+1})}\big(r^x_{LB,i+1,j+1} + r^x_{RB,i+1,j+1}\big) \\
&\quad - \frac{h_{i+1}}{(2\sigma_{tr,i+1,j+1}k_{j+1})}\big(r^y_{LB,i+1,j+1} + r^y_{LT,i+1,j+1}\big).
\end{aligned}$$

After the remaining three balance equations are similarly transformed, the fourth and final step is to sum the four balance equations. The currents cancel, giving an expression for the scalar fluxes at the vertices. The resulting right-hand side defines the operator for projecting a vector from the DFE trial space onto the CFE trial space. Presentation of the projection matrices is deferred for the time being. We simply write the standard lumped, vertex centered, five-point CFE diffusion equation stencil at vertex (i,j) with some source term $s_{i,j}$:

$$\begin{aligned}
&-\frac{1}{2}\left(\frac{D_{i-1,j-1}}{c_{i-1,j-1}} + \frac{D_{i-1,j}}{c_{i-1,j}}\right)\varphi_{i-1,j} - \frac{1}{2}\big(c_{i-1,j-1}D_{i-1,j-1} + c_{i,j-1}D_{i,j-1}\big)\varphi_{i,j-1} \\
&\quad + \frac{1}{4}\Bigg(c_{i-1,j-1}D_{i-1,j-1} + \frac{D_{i-1,j-1}}{c_{i-1,j-1}} + 2\tau_{i-1,j-1} + c_{i-1,j}D_{i-1,j} + \frac{D_{i-1,j}}{c_{i-1,j}} + 2\tau_{i-1,j} \\
&\quad + c_{i,j-1}D_{i,j-1} + \frac{D_{i,j-1}}{c_{i,j-1}} + 2\tau_{i,j-1} + c_{i,j}D_{i,j} + \frac{D_{i,j}}{c_{i,j}} + 2\tau_{i,j}\Bigg)\varphi_{i,j} \\
&\quad - \frac{1}{2}\left(\frac{D_{i,j-1}}{c_{i,j-1}} + \frac{D_{i,j}}{c_{i,j}}\right)\varphi_{i+1,j} - \frac{1}{2}\big(c_{i-1,j}D_{i-1,j} + c_{i,j}D_{i,j}\big)\varphi_{i,j+1} = s_{i,j},
\end{aligned} \qquad (3.32)$$

where $c_{i,j} = h_i/k_j$, $\tau_{i,j} = \sigma_{a,i,j}h_ik_j$, and $D_{i,j} = 1/(3\sigma_{t,i,j})$,

A vacuum condition is used when the continuous diffusion equations are used as a preconditioner and there is a vacuum or isotropic source boundary condition on the P$_1$ equations. Reflective conditions are used if reflection is specified for the P$_1$ equations. The matrix is SPD in the presence of either of these boundary conditions.

4. Preconditioning the Iterative Solution. We use a preconditioned Krylov iterative method to solve the nonsymmetric linear system $\mathbf{A}\bar{x} = \bar{b}$, assembled from the DFE discretization of the P$_1$ equations. Iterative solution techniques are preferred because $\mathbf{A}$ will sparse and direct methods are impractical for large problems. After extensive testing with several Krylov solvers, we found GMRES(m) to be the most robust and fastest converging transpose free, nonsymmetric Krylov method for this application [35]. We will not report results with other methods in this paper. An advantage of using Krylov solvers is that only matrix-vector products need to be computed. Compact matrix storage schemes can therefore be used to reduce memory requirements. This also makes specialized or parallel-distributed methods easy to implement.

Convergence can be improved by (left) preconditioning the equivalent linear system $\mathbf{M}^{-1}\mathbf{A}\bar{x} = \mathbf{M}^{-1}\bar{b}$, where the matrix $\mathbf{M}^{-1}$ in some sense approximates the inverse of $\mathbf{A}$. If $\mathbf{M}^{-1}\mathbf{A}$ is "close" to the identity, convergence is accelerated by clustering and/or compressing the eigenvalue spectrum of $\mathbf{A}$. The matrix $\mathbf{M}$ does not necessarily have to formed or stored nor does the inverse $\mathbf{M}^{-1}$ necessarily have to be computed. This is because the Krylov algorithms only require that a preconditioner return the preconditioned vector $\bar{z}$. This can be calculated by computing $\bar{z} = \mathbf{M}^{-1}\bar{r}$ or, equivalently, by solving the system $\mathbf{M}\bar{z} = \bar{r}$ for $\bar{z}$. This is referred to as the "action" of the preconditioner on a vector. Preconditioners based on the solution to another related, and simpler, linear system might also be used, which may in turn require preconditioning as well.

Preconditioning depends on finding an appropriate $\mathbf{M}$ whose action can be computed efficiently. The search for effective preconditioners can often be more of an art than a science. For instance, when the matrix represents a discretized set of of differential equations on some domain or in some regime, we can take advantage of the physics of the problem being solved to help identify effective preconditioners. Note that we have only preconditioned on the left. This is because we have found that our system of equations is scaled in such a way that right preconditioning can actually increase the condition number in some cases and left preconditioning is effective for a wide range of problems, at least for the preconditioner we will describe here.

As the optical thickness of the cells increases in transport problems that are highly diffusive, iterative convergence degrades and discontinuous solutions tend to become continuous at the cell interfaces. This observation led us to believe that a linear, *continuous* finite element discretization of the diffusion equation should be an effective preconditioner for the DFE discretization of the P_1 equations. The standard CFE diffusion equation is the basis of our preconditioning algorithm. A DFE vector must first first projected from the DFE trial space onto the lower dimensional CFE trial space. The CFE diffusion equations are then solved with this projected vector as a source and the solution is interpolated back to the DFE trial space. In thick, diffusive problems the CFE diffusion equations should be a good approximation to the DFE P_1 equations solution and the preconditioner should be effective. The method should be efficient because the CFE discretization involves only the scalar fluxes on the vertices.

We view the preconditioner as "solving" the system $\mathbf{M}\bar{z} = \bar{r}$ for $\bar{r}, \bar{z} \in \mathbb{R}^n$. The procedure is shown in Algorithm 1.

Algorithm 1. Two-Level Preconditioner

```
$\bar{z} \leftarrow 0$
$\bar{s} \leftarrow \bar{r} - \mathbf{A}\bar{z}$
$\bar{z} \leftarrow \bar{z} + \omega_1 \tilde{\mathbf{A}}^{-1}\bar{s}$
$\bar{s} \leftarrow \bar{r} - \mathbf{A}\bar{z}$
$\bar{z} \leftarrow \bar{z} + \mathbf{C}^{-1}\bar{s}$
$\bar{s} \leftarrow \bar{r} - \mathbf{A}\bar{z}$
$\bar{z} \leftarrow \bar{z} + \omega_2 \tilde{\mathbf{A}}^{-1}\bar{s}$
```

The bottom of the two-level algorithm consists of the operations that make up the *operator* $\mathbf{C}^{-1}$, that is, $\mathbf{C}^{-1} = \mathbf{Q}\mathbf{F}^{-1}\mathbf{P}$. The DFE P_1 equations have twelve unknowns per cell, so the total number of unknowns in the problem is $n = 12N_xN_y$. The CFE diffusion equation matrix is of lower dimension (rank) with unknowns located at the cell vertices such that $\mathbf{F} \in \mathbb{R}^{m\times m}$, where $m = (N_x + 1)(N_y + 1)$. It is therefore necessary to project from the higher dimensional DFE representation onto the lower dimensional CFE representation and interpolate back. The projection operator $\mathbf{P} \in \mathbb{R}^{m\times n}$ results from the manipulations described above in determining the CFE discretization of the diffusion equation from the discontinuous P_1 equations. The right-hand side in those expression is now represented by the vector $\bar{s}$ in Algorithm 1. The operator $\mathbf{Q} \in \mathbb{R}^{n\times m}$ interpolates the solution on the from the CFE representation on the vertices back onto the DFE representation. These operators will be discussed in greater detail below.

The matrix $\mathbf{F}$ is a relatively small, sparse, banded SPD matrix that can be (inexactly) solved efficiently with the CG algorithm. Currently it is preconditioned only with diagonal scaling. For larger problems and parallel implementations more sophisticated approaches like sparse approximate inverses [8] or a multigrid method [11] could be considered. We found that the time spent solving the "inner" CFE diffusion problem can influence on the optimality of the overall method. It is also possible that the convergence rate and the tolerance of the inner solution can affect the convergence of the "outer" iterations [9, 16]. As is commonly done with problems involving inner and outer iterations, the inner convergence tolerance could be scaled by the "size" of the outer residual to improve overall efficiency. We plan to investigate these issues in the future.

In Algorithm 1, the CFE diffusion solution is preceded and followed by smoothing operations, where the matrix $\tilde{\mathbf{A}}$ represents some simple approximation to $\mathbf{A}$. The first three steps of the preconditioning algorithm can be collapsed into a single operation, but we present it in this form to make clear the V-cycle nature of the two-level algorithm. Smoothing is necessary to avoid the possibility of encountering a vector in the null space of the operator $\tilde{\mathbf{C}}$, denoted by $N(\tilde{\mathbf{C}})$, which could possibly lead to false convergence. This is of concern because the null space has dimension $\dim(N(\tilde{\mathbf{C}})) = n - m$, which in our case can be quite large.

The choice of smoothers can influence the effectiveness of the preconditioner. In thick, diffusive problems we can simply choose $\tilde{\mathbf{A}} = \mathbf{I}$ where $\mathbf{I}$ is the $n \times n$ identity, which corresponds to a simple Richardson iteration. We found that something more sophisticated was needed for problems with thin or high aspect ratio cells. In this case we use block Jacobi smoothers, either in the form of block cell relaxations or x–line and y–line relaxations. Which of these is best depends on the problem. We will determine this not only by the effectiveness and efficiency of the relaxations but also by their potential for improving the overall solution time when applied to large problems or implemented in parallel.

The block cell Jacobi relaxation uses $\tilde{\mathbf{A}} = \mathbf{B}_D$, where $\mathbf{B}_D \in \mathbb{R}^{n\times n}$ is the block diagonal of $\mathbf{A}$. Each block consists is 12×12, corresponding to the unknowns on cell (i, j). In the notation of the previous section,

$$\mathbf{B}_D = \mathrm{diag}(\mathbf{A}_{i,j}), \quad i = 1, N_x, j = 1, N_y. \tag{4.1}$$

This smoother is best suited for parallel, unstructured, three dimensional applications but is not very effective for problems with thin cells or high aspect ratio cells. This can be addressed with an x and y line relaxation smoother. In that case, $\tilde{\mathbf{A}} = \mathbf{B}_{xy}$, where

$$\mathbf{B}_{xy}^{-1} = \frac{1}{2}\mathbf{B}_x^{-1} + \frac{1}{2}\mathbf{B}_y^{-1}. \tag{4.2}$$

The matrix $\mathbf{B}_x$ is an $N_y \times N_y$ block diagonal matrix, each block consisting of an $N_x \times N_x$ block tridiagonal matrix (of 12×12 blocks). Similarly, the matrix $\mathbf{B}_y$ is

$N_y \times N_y$ block diagonal, each block being $N_x \times N_x$ block tridiagonal. Using the notation of the previous section,

$$\mathbf{B}_x = \text{diag}(\mathbf{B}_j), \quad j = 1, N_y, \tag{4.3}$$

$$\mathbf{B}_y = \text{diag}(\mathbf{B}_i), \quad i = 1, N_x, \tag{4.4}$$

where

$$\mathbf{B}_j = \begin{bmatrix} \mathbf{A}_{1,j} & \mathbf{X}_j & & & \\ \mathbf{W}_j & \mathbf{A}_{2,j} & \mathbf{X}_j & 0 & \\ & & \ddots & & \\ & 0 & \mathbf{W}_j & \mathbf{A}_{(N_x-1),j} & \mathbf{X}_j \\ & & & \mathbf{W}_j & \mathbf{A}_{N_x,j} \end{bmatrix}$$

$$\mathbf{B}_i = \begin{bmatrix} \mathbf{A}_{i,1} & \mathbf{Z}_i & & & \\ \mathbf{Y}_i & \mathbf{A}_{i,2} & \mathbf{Z}_i & 0 & \\ & & \ddots & & \\ & 0 & \mathbf{Y}_i & \mathbf{A}_{i,(N_y-1)} & \mathbf{Z}_i \\ & & & \mathbf{Y}_i & \mathbf{A}_{i,N_y} \end{bmatrix}.$$

The block line smoother is clearly going to be less efficient than the block cell smoother, in terms of both memory storage and execution time. On the other hand, the block line smoother is more effective. There could problems for which the increased cost per iteration is outweighed by a reduction in iteration counts. However, the block line smoothers are more difficult to implement in parallel because lines of cells would have to be divided among processors. Furthermore, on unstructured three dimensional meshes, "lines" can only be generated based on the connectivity of the mesh. A compromise might be to extend the block cell relaxations to include nearby cells to improve convergence in the case of thin or high aspect ratio cells. This could also be extended to include overlapping blocks like an additive Schwarz algorithm [27].

Note that we have allowed the possibility of damping the pre- and post-smoothing relaxations. When $\tilde{\mathbf{A}} = \mathbf{B}_D$ we take $\omega_1 + \omega_2 = 1$ and when $\tilde{\mathbf{A}} = \mathbf{B}_{xy}$ we use $\omega_1 = \omega_2 = 1$. As it stands, the preconditioner can be used as stand-alone iterative solution algorithm whose convergence and stability can be easily analyzed. Although we will not discuss the details of this analysis, it is what led us to use line relaxations and to our choices for ω_1 and ω_2.

We will now discuss the details of projection and interpolation operators. The projection operators arise from the derivations of the CFE discretization of the diffusion equation with some DFE vector as a source term. The projection operator

$P \in \mathbb{R}^{m\times n}$ can be written by defining the following row vectors in $\mathbb{R}^4$,

$$\begin{aligned}
\bar{\nu}^0_{LB,i,j} &= [1,\ 0,\ 0,\ 0], \quad \bar{\nu}^0_{RB,i,j} = [0,\ 1,\ 0,\ 0],\\
\bar{\nu}^0_{LT,i,j} &= [0,\ 0,\ 1,\ 0], \quad \bar{\nu}^0_{RT,i,j} = [0,\ 0,\ 0,\ 1],\\
\bar{\nu}^x_{B,i,j} &= \frac{k_j}{2}\left(\sigma_{t,i,j}h_i\right)^{-1}[1,\ 1,\ 0,\ 0], \quad \bar{\nu}^x_{T,i,j} = \frac{k_j}{2}\left(\sigma_{t,i,j}h_i\right)^{-1}[0,\ 0,\ 1,\ 1]\\
\bar{\nu}^y_{L,i,j} &= \frac{h_i}{2}\left(\sigma_{t,i,j}k_j\right)^{-1}[1,\ 0,\ 1,\ 0], \quad \bar{\nu}^y_{R,i,j} = \frac{h_i}{2}\left(\sigma_{t,i,j}k_j\right)^{-1}[0,\ 1,\ 0,\ 1],
\end{aligned}$$

to construct the four row vectors, each in $\mathbb{R}^{12}$,

$$\begin{aligned}
\bar{\nu}_{LB,i,j} &= \left[\bar{\nu}^0_{LB,i,j}, -\bar{\nu}^x_{B,i,j}, -\bar{\nu}^y_{L,i,j}\right], \quad \bar{\nu}_{RB,i,j} = \left[\bar{\nu}^0_{RB,i,j},\ \ \bar{\nu}^x_{B,i,j}, -\bar{\nu}^y_{R,i,j}\right],\\
\bar{\nu}_{LT,i,j} &= \left[\bar{\nu}^0_{LT,i,j}, -\bar{\nu}^x_{T,i,j},\ \ \bar{\nu}^y_{L,i,j}\right], \quad \bar{\nu}_{RT,i,j} = \left[\bar{\nu}^0_{RT,i,j},\ \ \bar{\nu}^x_{T,i,j},\ \ \bar{\nu}^y_{R,i,j}\right],
\end{aligned}$$

which are used to define the matrices $\mathbf{P}_{B,j}, \mathbf{P}_{T,j} \in \mathbb{R}^{(N_x+1)\times 12N_x}$,

$$\mathbf{P}_{B,j} = \begin{bmatrix} \bar{\nu}_{LB,1,j} & 0 & & \\ \bar{\nu}_{RB,1,j} & \bar{\nu}_{LB,2,j} & & \\ & \ddots & \ddots & \\ & & \bar{\nu}_{RB,N_x-1,j} & \bar{\nu}_{LB,N_x,j} \\ & & 0 & \bar{\nu}_{RB,N_x,j} \end{bmatrix}$$

$$\mathbf{P}_{T,j} = \begin{bmatrix} \bar{\nu}_{LT,1,j} & 0 & & \\ \bar{\nu}_{RT,1,j} & \bar{\nu}_{LT,2,j} & & \\ & \ddots & \ddots & \\ & & \bar{\nu}_{RT,N_x-1,j} & \bar{\nu}_{LT,N_x,j} \\ & & 0 & \bar{\nu}_{RT,N_x,j} \end{bmatrix}$$

such that the projection matrix is the $(N_y+1)\times N_y$ block matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{B,1} & 0 & & \\ \mathbf{P}_{T,1} & \mathbf{P}_{B,2} & & \\ & \ddots & \ddots & \\ & & \mathbf{P}_{T,N_y-1} & \mathbf{P}_{B,N_y} \\ & & 0 & \mathbf{P}_{T,N_y} \end{bmatrix}. \tag{4.5}$$

The interpolation operator is represented by the matrix $Q \in \mathbb{R}^{n\times m}$. We call our interpolation an "update" because the components of the vector corresponding to the discontinuous scalar fluxes for the cells surrounding a vertex of the mesh are all corrected with the same value of the continuous diffusion equation solution at that vertex. The currents are not corrected. Define the following vectors in $\mathbb{R}^{12}$,

$$\bar{\mu}_{LB,i,j} = [1,\ 0,\ 0,\ 0,\ 0,\ldots 0]^T, \quad \bar{\mu}_{RB,i,j} = [0,\ 1,\ 0,\ 0,\ 0,\ldots 0]^T,$$
$$\bar{\mu}_{LT,i,j} = [0,\ 0,\ 1,\ 0,\ 0,\ldots 0]^T, \quad \bar{\mu}_{RT,i,j} = [0,\ 0,\ 0,\ 1,\ 0,\ldots 0]^T,$$

to construct the matrices $\mathbf{Q}_{B,j}, \mathbf{Q}_{T,j} \in \mathbb{R}^{12N_x\times(N_x+1)}$,

$$\mathbf{Q}_{B,j} = \begin{bmatrix} \bar{\mu}_{LB,1,j} & \bar{\mu}_{RB,1,j} & & & & \\ 0 & \bar{\mu}_{LB,2,j} & \bar{\mu}_{RB,2,j} & & & \\ & & \ddots & \ddots & & \\ & & & \bar{\mu}_{LB,N_x-1,j} & \bar{\mu}_{RB,N_x-1,j} & 0 \\ & & & & \bar{\mu}_{LB,N_x,j} & \bar{\mu}_{RB,N_x,j} \end{bmatrix}$$

$$\mathbf{Q}_{T,j} = \begin{bmatrix} \bar{\mu}_{LT,1,j} & \bar{\mu}_{RT,1,j} & & & & \\ 0 & \bar{\mu}_{LT,2,j} & \bar{\mu}_{RT,2,j} & & & \\ & & \ddots & \ddots & & \\ & & & \bar{\mu}_{LT,N_x-1,j} & \bar{\mu}_{RT,N_x-1,j} & 0 \\ & & & & \bar{\mu}_{LT,N_x,j} & \bar{\mu}_{RT,N_x,j} \end{bmatrix}$$

such the interpolation matrix is the $N_y \times (N_y + 1)$ block matrix

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{B,1} & \mathbf{Q}_{T,1} & & & & \\ 0 & \mathbf{Q}_{B,2} & \mathbf{Q}_{T,2} & & & \\ & & \ddots & \ddots & & \\ & & & \mathbf{Q}_{B,N_y-1} & \mathbf{Q}_{T,N_y-1} & 0 \\ & & & & \mathbf{Q}_{B,N_y} & \mathbf{Q}_{T,N_y} \end{bmatrix}. \tag{4.6}$$

Multigrid or other multi-level methods often specify that the projection and interpolation matrices are transposes of one another. We found that if enforcing symmetry by choosing one of the matrices to be the transpose of the other destroys the effectiveness of our preconditioner.

Finally, note that we derived another interpolation based on the assumption that partial currents on the faces of the cells can be written in terms of the CFE scalar fluxes which may then be related to the discontinuous P_1 equation quantities using an approximation to the partial currents [1, 28, 37]. This interpolation is, however, difficult to derive and complicated to evaluate especially in higher dimensions. But we found that the simple interpolation presented here performed nearly as well as these more expensive interpolation anyway. The interested reader can consult [28] for more details.

5. Analysis and Numerical Experiments.

Evidence of the effectiveness of our preconditioning algorithm will be presented in the form of a Fourier analysis along with measurements of the performance of an actual implementation. The methods were implemented and executed on serial machines. We will indicate directions for application of the algorithm to large problems on parallel computer platforms suggested by results from the serial test program.

The anticipated convergence rates for the nonsymmetric Krylov methods are difficult to characterize and remain open questions [17]. If the preconditioned matrix were positive definite, that is, if the matrix has an SPD symmetric part, then we could bound the convergence rate of GMRES(m) based on the distribution of the eigenvalues of the symmetric part and the singular values of the preconditioned system [30]. Unfortunately, in our case $\mathbf{A}$ is nonsymmetric positive definite, but we found that $\mathbf{M}^{-1}\mathbf{A}$ is not, in general, nonsymmetric positive definite. We are thus left without an analytical bound on the convergence of GMRES(m) as well as the possibility that the method might stagnate (though we have not seen this occur in practice). On the other hand, we have found that we can get an idea of the relative convergence rates that can be expected among problems with varying cell shapes and thicknesses by calculating the condition number based on the ratio of the maximum and minimum singular values of the preconditioned system for some small, model problems. These condition numbers can be also be compared to those estimated by a Fourier analysis.

One issue of efficiency of the implementation should be noted. Both of the block Jacobi smoothers were computed, inverted, and stored ahead of time, before the outer iterations started. This storage overhead could be eliminated at the cost of additional computations at every outer iteration, but this is extremely inefficient, especially for the block line smoother.

5.1. Condition Numbers and Convergence of GMRES(m).

We have computed the full matrix for the preconditioned system of a problem with constant material properties, keeping the number of mesh cells in each dimension constant, $N_x = N_y = 10$. We will refer to this as the "fixed-size problem". The number of unknowns is $n = 1200$. By varying the physical dimensions of the problem domain we can see how the condition number and iterative convergence depends on mesh thickness and aspect ratio. The total cross section is 1cm^{-1} and the scattering ratio is 0.99990. The dimensions of the problem are taken to be $10\Delta x(\text{cm}) \times 10\Delta y(\text{cm})$. The ratio of the largest and smallest singular values gives the condition number in the 2–norm. The measured values for problems without preconditioning as well as for preconditioned problems are shown below. In all the tabulated results presented in this section, numbers written in the form $x.x(y)$ are to be interpreted as $x.x \cdot 10^y$.

It is also possible to estimate the condition number of the preconditioned system with a discrete Fourier analysis. We assume a discrete Fourier ansatz of the form

$fe^{i\lambda_x mh}e^{i\lambda_y nk}$ for each the twelve DFE unknowns on a cell (m,n) (a different f for each unknown on the cell) in terms of the cell width h and k assuming that they and the material properties are constant throughout. We then do the same on the cell vertex (m,n) with the discretized continuous scalar flux having the form $ge^{i\lambda_x mh}e^{j\lambda_y nk}$, and insert the two ansatz into the equations. The result is a 12×12 matrix for the discrete Fourier coefficients corresponding to the (preconditioned) equations. We can then compute the singular values of this matrix to estimate the condition number.

In Tables 1–4 both measured condition numbers and the results of the Fourier analysis are tabulated. Each entry in the table for a Δx, Δy combination lists the measured condition number with vacuum boundary conditions, the measured condition with reflective boundary conditions, and the estimate from the Fourier analysis. In essence, the Fourier analysis estimates the condition number for a problem with an infinite number of identical cells. The measured condition numbers are, of course, for problems with a finite number of cells, including the effects of boundaries. Thus it makes sense that the Fourier analysis estimates are in better agreement with the measured condition numbers for reflective boundary conditions because the physical situation corresponds more closely to the assumptions made in the Fourier analysis. This is especially true for "artificial" problems such as these, which were very thin in one of the dimensions in order to see the effects of high aspect ratios. A more meaningful comparison would be to compute the condition numbers for problems with fixed dimensions and increasing the numbers of cells in one or the other dimension to construct cells with high aspect ratios. Unfortunately, this is a computationally difficult problem because of the large number of unknowns. A 10×10 cell problem (1200 unknowns) was a practical limit.

We also computed the solution to the same problems using with GMRES(20) with our two-level preconditioner. We used vacuum boundary conditions on the four faces of the problem. An isotropic source of randomly distributed strength over $[0, 1]$ particles/cm^3 s is specified throughout the problem. The convergence criterion is $\|\bar{r}^k\|_2 \leq 10^{-5}\|\bar{b}\|_2$, where $\bar{r}^k$ is the recursively computed GMRES residual and $\bar{b}$ is the source vector. The continuous finite element diffusion solution in the preconditioner is computed using CG to a tolerance of $\|\bar{r}^k\|_2 \leq 10^{-11}\|\bar{z}\|_2$, where $\bar{r}^k$ is the "inner" CG residual and $\bar{z}$ is a *projected* DFE vector. It is possible to relax the inner convergence criteria while maintaining the effectiveness of the preconditioner to make the overall algorithm more efficient. But this strict tolerance was imposed on the inner iterations to ensure there is no effect on the outer iteration convergence due to accumulation of error from the preconditioner. The number of iterations are tabulated together with CPU time in Tables 5–8. CPU time is measured in seconds on a single user SGI Octane. It includes the (negligible) matrix setup time. If convergence was not reached in 2000 iterations then we list the final residual norm in brackets.

The first observation to make from the condition number calculations is that our preconditioner, regardless of the smoother being used, effectively improves the convergence of GMRES compared to no preconditioning at all. It is most effective for problems with cells that have an optical thickness greater than one in both dimensions. If the cells have an aspect ratio, then the effectiveness of the preconditioner is reduced and the smoother plays an increasingly important role. If the cells are optically very thin in either of the two dimensions, the preconditioner loses effectiveness; both block relaxation smoothers help in this case. If the cells are very optically thick in one dimension and very optically thin in the other, then the line relaxation

	Δy						
Δx	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}

10^{-4}	1.8(5)
	3.3(12)
	3.4(12)
		1.0(5)
		3.3(11)
		3.4(11)
			2.9(4)
			3.3(10)
			3.4(10)
				2.0(3)
				3.3(9)
				3.4(9)
					6.5(1)
					4.9(8)
					4.9(8)
						1.4(2)
						2.3(8)
						2.3(8)
							1.3(3)
							2.2(8)
							2.2(8)

10^{-3}		1.8(4)
		3.3(10)
		3.3(10)
			7.7(3)
			3.3(9)
			3.3(9)
				9.5(2)
				3.4(8)
				3.4(8)
					6.4(1)
					4.9(7)
					4.9(7)
						1.4(2)
						2.3(7)
						2.3(7)
							1.3(3)
							2.2(7)
							2.2(7)

10^{-2}			1.8(3)
			3.3(8)
			3.3(8)
				3.6(2)
				3.4(7)
				3.4(7)
					5.7(1)
					4.9(6)
					4.9(6)
						1.3(2)
						2.3(6)
						2.3(6)
							1.2(3)
							2.2(6)
							2.2(6)

10^{-1}				2.3(2)
				3.4(6)
				3.4(6)
					6.5(1)
					4.9(5)
					4.9(5)
						1.1(2)
						2.3(5)
						2.3(5)
							1.1(3)
							2.2(5)
							2.2(5)

10^{0}					1.1(2)
					5.7(4)
					5.7(4)
						1.4(2)
						2.3(4)
						2.3(4)
							1.1(3)
							2.2(4)
							2.2(4)

10^{1}						5.3(2)
						3.8(3)
						3.8(3)
							1.0(3)
							2.3(3)
							2.3(3)

10^{2}							3.7(2)
							3.8(2)
							3.9(2)

Table 1. Condition numbers for the fixed-size problem *without* preconditioning. Entries are for (1) vacuum boundary conditions, (2) reflective boundary conditions, (3) Fourier analysis estimate.

smoother is most effective. If the cells are optically thin in both dimensions, the preconditioner begins to lose effectiveness. The extent of this degradation is also strongly affected by the smoother. Comparing computational effort with the condition number calculations shows that there is a rough correspondence between the effectiveness of the preconditioner and condition number. This is potentially very useful. A Fourier analysis could predict the relative improvements in convergence that can be expected among various other possible smoothers or preconditioning algorithms in the future.

To summarize, the preconditioner is very effective and significantly improves convergence of GMRES. Performance in different limits of the cell optical thickness in each dimension can be improved to a greater or lesser extent by the choice of smoother. One might also question whether the relaxations alone, without GMRES, are effective preconditioners. We will report on this in the future, but we simply state here that the continuous finite element diffusion solution is essential for effective preconditioning, especially in the thick transport problems in which we plan to implement solution of the discontinuous P_1 equations.

5.2. A Numerical Experiment. We will now present the results of some numerical experiments to measure how computational effort scales with problem size. The problem is a variation of the two-dimensional iron-water problem used as an example problem for neutron transport many times over the years (see [19], for example). It contains several types of regions, some diffusive and some not. The coarsest mesh is illustrated in Fig. 2. It consists of a 30cm $\times$ 30cm square, heterogeneous system containing iron and water. The left and bottom faces are reflective with vacuum boundary conditions on the right and top faces. A unit distributed source is located

	Δy						
Δx	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}

10^{-4}	1.6(5)
	1.6(5)
	1.2(6)
		5.9(5)
		6.0(5)
		7.2(5)
			5.5(5)
			5.6(5)
			5.6(5)
				9.0(4)
				9.0(4)
				9.1(4)
					3.4(4)
					5.4(4)
					5.5(4)
						1.4(5)
						4.3(5)
						4.3(5)
							1.4(6)
							6.3(7)
							3.0(6)

10^{-3}		1.5(4)
		1.6(4)
		4.6(4)
			2.1(4)
			2.2(4)
			2.2(4)
				8.4(3)
				8.5(3)
				8.6(3)
					3.4(3)
					5.4(3)
					5.5(3)
						1.4(4)
						4.3(4)
						4.3(4)
							1.4(5)
							3.0(5)
							3.0(5)

10^{-2}			1.2(3)
			1.2(3)
			1.5(3)
				5.8(2)
				6.0(2)
				6.0(2)
					3.4(2)
					5.4(2)
					5.4(2)
						1.4(3)
						4.3(3)
						4.3(3)
							1.4(4)
							3.0(4)
							3.0(4)

10^{-1}				4.7(1)
				5.3(1)
				5.4(1)
					3.1(1)
					5.1(1)
					5.2(1)
						1.4(2)
						4.3(2)
						6.2(2)
							1.3(3)
							3.0(3)
							1.1(4)

10^{0}					5.2(0)
					5.3(0)
					5.3(0)
						3.5(2)
						6.2(1)
						6.4(1)
							7.7(2)
							1.1(3)
							1.3(3)

10^{1}						2.3(1)
						2.4(1)
						2.4(1)
							1.5(2)
							1.7(2)
							1.7(2)

10^{2}							1.4(2)
							1.5(2)
							1.5(2)

Table 2. Condition Numbers for the fixed-size Problem using Algorithm 1 with Richardson smoothing. Entries are for (1) vacuum boundary conditions, (2) reflective boundary conditions, (3) Fourier analysis estimate.

in the 12cm × 12cm lower left corner. For water, the total cross section is 3.3333cm^{-1} and scattering ratio 0.9941 and for iron the total cross section is 1.3333cm^{-1} and scattering ratio 0.8308. Larger problems with greater numbers of unknowns and correspondingly thinner cells were generated by taking multiples of the "base" discretization of a $N_x = 15 \times N_y = 15$ mesh. We chose a convergence tolerance of $\|\bar{r}^k\| \leq 10^{-5}\|\bar{b}\|$ and 5000 maximum iterations. The diagonally scaled inner CG iterations were computed to a tolerance of $\|\bar{r}^k\| \leq 10^{-11}\|\bar{z}\|$ where $\bar{r}^k$ is the CG residual and $\bar{z}$ is a *projected* DFE vector. The computations were carried out on a single SGI Origin2000 compute server.

In the first series of problems we used GMRES(20) to solve the iron-water problem as the number of cells in the base problem is scaled by some factor f in the x and y dimensions simultaneously. The number of unknowns is $12 \cdot 15^2 \cdot f^2$ and $f = 1, 2, 4, \ldots, 20$. However, even with a very large memory of 15GB and compressed storage of the block Jacobi matrices the largest problem using the block line smoother was limited to $f = 14$ (a total number of 529,200 unknowns). The number of iterations and CPU time for the solution of the preconditioned problem are shown in Fig. 3 for three different smoothers. We already indicated that the execution time needed to extract and compute the block line matrices at every invocation of the preconditioner was prohibitive. The only practical solution was to invert and store these matrices ahead of time. However, the high setup times required to invert and store these matrices added significantly to the total execution time. This may be seen in Fig. 3(b) where we plotted the time just for the iterative solution using the block line smoother in addition to the total time. Once the problem size was greater than $f^2 = 64$ the fraction of time spent setting up the block line matrices was constant, about 90% of the

	Δy						
Δx	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
10^{-4}	1.0(4) 1.1(4) 1.1(4)	7.6(3) 8.5(3) 1.5(4)	1.0(3) 1.2(3) 1.6(4)	1.4(2) 2.6(3) 1.6(4)	3.1(1) 2.3(4) 2.0(4)	3.0(1) 2.2(5) 1.9(5)	3.0(1) 2.2(6) 1.9(6)
10^{-3}		1.0(3) 1.1(3) 1.1(3)	7.6(2) 8.4(2) 1.5(3)	9.7(2) 2.6(2) 1.6(3)	3.1(1) 2.3(3) 2.0(3)	3.0(1) 2.2(4) 1.9(4)	3.0(1) 2.2(5) 1.9(5)
10^{-2}			1.0(2) 1.1(2) 1.1(2)	7.4(1) 8.2(1) 1.4(2)	2.6(1) 2.2(2) 2.0(2)	2.5(1) 2.2(3) 1.9(3)	2.5(1) 2.2(4) 1.9(4)
10^{-1}				1.2(1) 1.3(1) 1.2(1)	1.1(1) 2.0(1) 1.8(1)	2.3(1) 2.1(2) 1.9(2)	2.6(1) 2.2(3) 1.9(3)
10^{0}					2.8(0) 3.4(0) 2.9(0)	1.4(1) 1.9(1) 1.8(1)	5.2(1) 2.1(2) 1.8(2)
10^{1}						2.7(0) 3.3(0) 2.8(0)	1.7(1) 1.7(1) 1.7(1)
10^{2}							2.5(0) 2.7(0) 2.5(0)

Table 3. Condition numbers for the fixed-size problem using Algorithm 1 block cell relaxations. Entries are for (1) vacuum boundary conditions, (2) reflective boundary conditions, (3) Fourier analysis estimate.

total execution time. The setup times for the block cell preconditioner are negligible compared to the overall execution times, even for the largest problems and there is no additional storage or setup required for the Richardson smoother. Therefore, only total execution times are shown in Fig. 3(b) for the results with these smoothers.

The next figure, Fig. 4, shows a plot of floating point operations (FLOP) per unknown, including setup time, as a function of the problem size for the same problem. If such a plot were roughly constant, the solution method could be considered optimal. They were measured on the ORIGIN 2000 machines using the built-in counter (madds count as one floating point instruction on this machine). Under this scaling, the method does not appear to optimal which is likely a consequence of the diagonally scaled inner CG iteration, which does not scale optimally.

With these results in mind, we consider a different problem size scaling. Rather than simply increasing the number of unknowns, we will fix the level of discretization in terms of cell optical thicknesses by scaling the size of the physical domain of the base problem at the same time we scale the number of unknowns in the problem. In addition to increasing the number of unknowns in each dimension by the factor $f = 1, 2, 4, \ldots, 20$ as before, we also scale the physical x and y dimensions of the square domain of the base problem by the same factor. This keeps the cell optical thickness constant, so we call these "constant scaling" problems. The rest of the problem parameters are the same as before. The high setup costs associated with the block line relaxations seen in the previous set of problems and the potential difficulties of implementing them in parallel or on three dimensional unstructured meshes lead us to conclude that the block line smoother is noncompetitive. We will not include measurements using that smoother for these scalings.

	Δy						
Δx	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}

10^{-4}	9.5(3)
	9.7(3)
	6.3(4)
		1.4(4)
		1.6(4)
		1.6(4)
			2.1(3)
			4.7(3)
			1.8(6)
				6.7(1)
				1.1(2)
				1.0(2)
					2.6
					3.4
					3.1
						2.1
						2.5
						2.3
							2.1
							2.5
							2.3

10^{-3}		9.1(2)
		9.3(2)
		2.0(3)
			4.4(2)
			6.2(2)
			4.2(3)
				4.6(1)
				9.0(1)
				8.3(1)
					2.6
					3.4
					3.1
						2.1
						2.5
						2.3
							2.1
							2.5
							2.3

10^{-2}			6.2(1)
			6.5(1)
			6.9(1)
				1.5(1)
				3.1(1)
				2.9(1)
					2.3
					3.3
					3.0
						2.1
						2.4
						2.3
							2.1
							2.4
							2.3

10^{-1}				4.5(0)
				5.3(0)
				4.8(0)
					2.0
					2.7
					2.4
						2.0
						2.3
						2.2
							2.0
							2.4
							2.2

10^{0}					1.5 1.8 1.6	1.8 2.1 1.9	1.9 2.1 1.9
10^{1}						1.5 1.9 1.5	1.8 2.1 1.8
10^{2}							1.5 1.8 1.5

Table 4. Condition numbers for the fixed-size problem using Algorithm 1 with block line relaxations. Entries are for (1) vacuum boundary conditions, (2) reflective boundary conditions, (3) Fourier analysis estimate.

	Δy						
Δx	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
10^{-4}	[8.8(-1)]†	[9.3(-1)]†	[9.4(-1)]†	808/3.0	182/0.7	231/0.9	1535/5.6
10^{-3}		[8.8(-1)]†	[9.3(-1)]†	810/3.0	183/0.2	218/0.9	[3.7(-4)]†
10^{-2}			1155/4.2	695/2.6	182/0.7	212/0.8	[6.0(-5)]†
10^{-1}				302/1.2	158/0.6	168/0.7	873/3.2
10^{0}					85/0.4	109/0.4	240/0.9
10^{1}						149/0.6	178/0.7
10^{2}							51/0.2

Table 5. Number of GMRES(20) iterations and CPU time for the fixed-size problem with no preconditioning.
†Did not converge in 2000 iterations, final residual in brackets.

In Figure 5 we show the number of iterations and FLOP count per unknown for the constant scalings. In this case, not only does the problem scale optimally with the number of unknowns but the total computational effort is significantly lower. This is an encouraging result if we know we are going to solve problems with a very large numbers of cells on a mesh with well shaped cells that are not too thin. This also indicates the results for the original scaling are determined by the mesh cell size. Furthermore, with a fixed mesh cell size there is not a large difference in either the number of iterations or the computational effort between the two smoothers.

More will be said regarding these scalings later, after we first compare the precon-

	Δy						
Δx	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
10^{-4}	[4.3(0)]†	[3.8(-1)]†	[5.2(-1)]†	[3.0(-3)]†	[4.2(-4)]†	[4.1(-5)]†	[4.1(-5)]†
10^{-3}		[8.2(-3)]†	[7.9(-4)]†	[7.2(-4)]†	[5.7(-5)]†	[1.8(-4)]†	[7.1(-5)]†
10^{-2}			613/25	547/41	548/27	1297/43	[6.37(-5)]†
10^{-1}				102/4.1	92/5.9	235/9.0	815/26
10^{0}					23/0.9	43/2.2	179/6.2
10^{1}						15/0.6	40/1.7
10^{2}							15/0.5

Table 6. Number of GMRES(20) iterations and CPU time for the fixed-size problem with Algorithm 1 and Richardson smoothing.
†Did not converge in 2000 iterations, final residual in brackets.

	Δy						
Δx	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
10^{-4}	68/5.9	[2.2(-5)]†	151/18	37/3.3	20/1.6	16/1.2	13/1.0
10^{-3}		57/4.9	292/34	51/5.9	25/2.1	18/1.4	15/1.1
10^{-2}			44/3.8	72/8.7	27/2.6	19/1.5	17/1.3
10^{-1}				25/2.2	24/2.7	19/1.6	16/1.3
10^{0}					11/1.0	12/1.2	11/0.9
10^{1}						6/0.6	10/0.9
10^{2}							4/0.4

Table 7. Number of GMRES(20) iterations and CPU time for the fixed-size problem using Algorithm 1 and block cell relaxations.
†Did not converge in 2000 iterations, final residual in brackets.

	Δy						
Δx	10^{-4}	10^{-3}	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
10^{-4}	195/113	253/157	39/24	10/6.1	8/4.9	7/4.3	5/3.3
10^{-3}		97/57	66/41	14/8.9	8/5.1	7/4.4	5/3.2
10^{-2}			42/25	19/12	8/5.1	7/4.4	5/3.3
10^{-1}				14/8.3	7/4.6	6/3.9	5/3.3
10^{0}					6/3.9	5/3.4	4/2.8
10^{1}						4/2.8	4/2.8
10^{2}							3/2.3

Table 8. Number of GMRES(20) iterations and CPU time for the fixed-size problem with Algorithm 1 and block line relaxations).

ditioned solution of the iron-water problem to that without preconditioning. This will give an idea of the effectiveness of the preconditioner, independent of computational computational efficiency. Because we found that GMRES(20) converges very slowly for this problem without preconditioning, the effectiveness of the preconditioners can be judged by comparing the convergence rates of the various methods. We define the

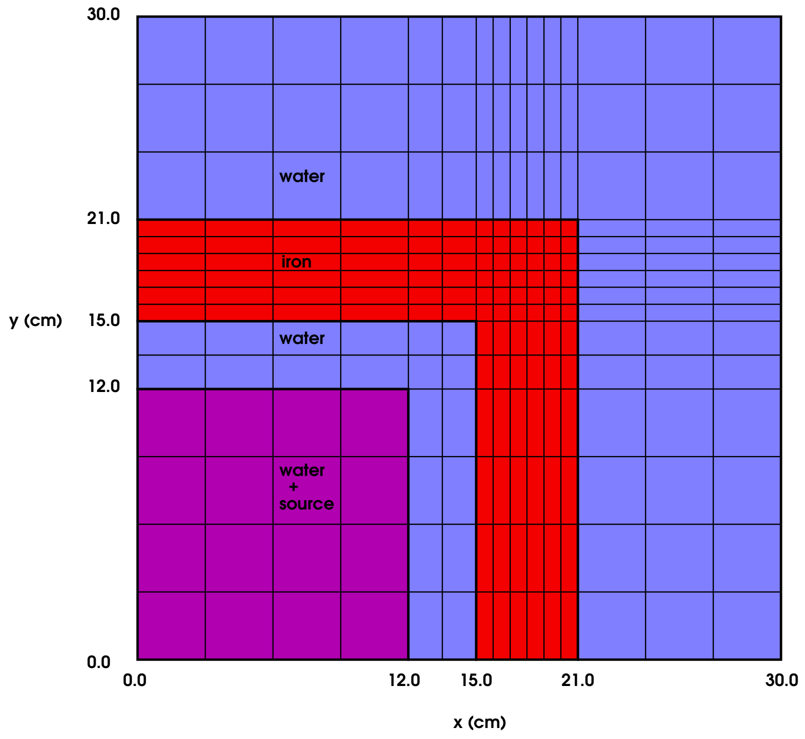


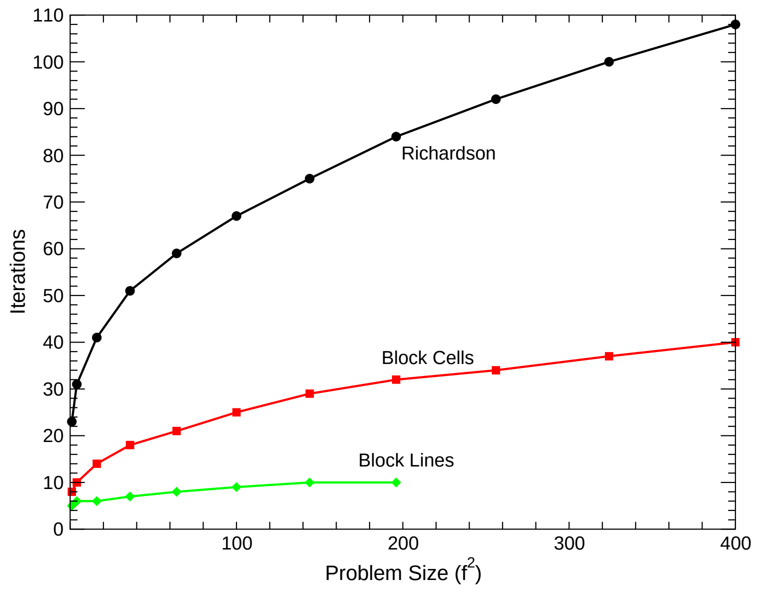
Fig. 2. Iron-water problem "base" mesh.

convergence rate by taking the average over all iterations k of

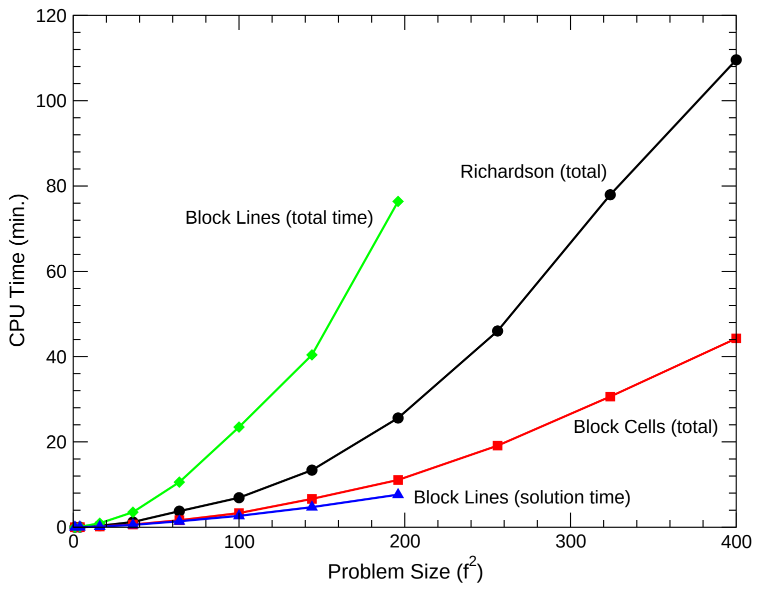
$$\left[-\log_{10}\left(\frac{\|r^{k+1}\|}{\|r^k\|}\right)\right]^{-1},$$

where $\|r^k\|$ is the norm of the computed GMRES(20) residual at iteration k. This represents the expected number of iterations needed to decrease the residual by one order of magnitude (one decimal digit). The average is used because convergence curves are not necessarily straight lines. The first three iterations are excluded from this average because we found that the the convergence curves "settled down" after the first few iterations.

In Table 9 we display the measured convergence rate as a function of the problem size factor f^2 for the original and the constant scalings. The outstanding effectiveness of the preconditioner is apparent. The effectiveness is especially impressive for the original scalings. With preconditioning, just a handful of iterations improves the solution by one decimal place. When the problem is not preconditioned, however, the constant scalings exhibit a constant convergence rate while the convergence rate increases in the original scalings. Taken together with the fact that the original scalings are not optimal, this implies that the number CG inner iterations is increasing with decreasing mesh width. This is probably related to the increasing condition number of the preconditioned system seen in the Fourier analysis and measurements shown earlier. The CFE diffusion discretization is closely related to 5–point finite difference stencils for second order equations. It is well known that such discretizations suffer from condition numbers that are inversely proportional to mesh sizes. It is possible,



3(a) Iteration count.



3(b) CPU time.

Fig. 3. Number of iterations and CPU time for the iron-water problem as a function of problem size. Three different smoothing relaxations are shown.

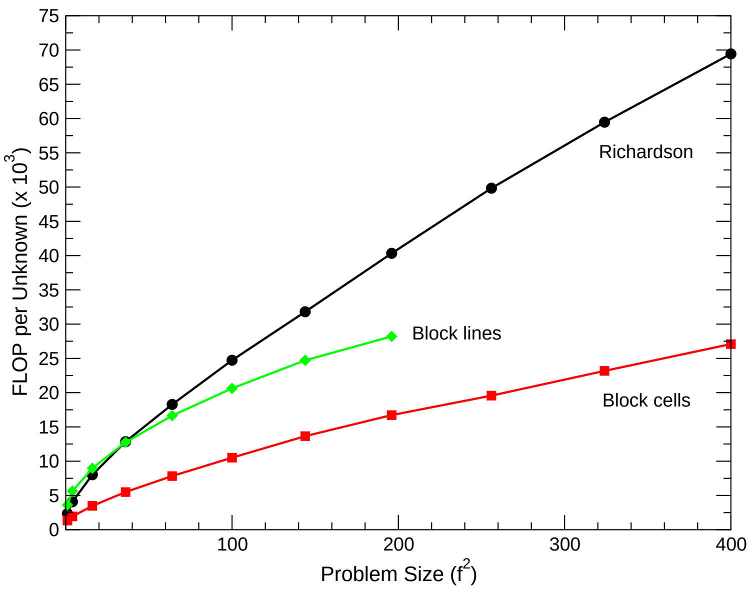


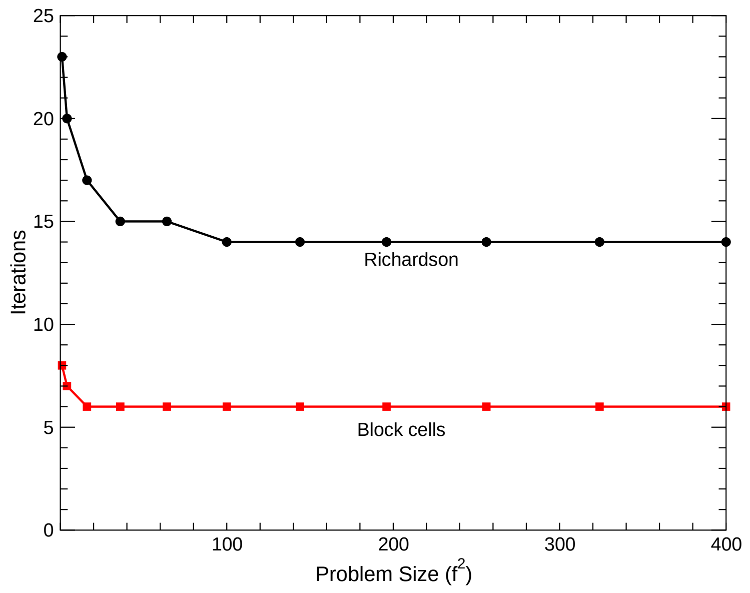
Fig. 4. Floating point operations (FLOP) per unknown as a function of problem size for three different smoothers.

Problem	Original Scaling				Constant Scaling		
Size (f^2)	NP[a]	R[b]	BC[c]	BL[d]	NP	R	BC
1	16.9	4.57	1.55	0.84	16.9	4.57	1.55
4	40.2	5.76	1.78	0.86	22.2	4.21	1.50
16	123	7.23	2.35	0.87	23.9	3.55	1.52
36	204	8.80	2.94	0.96	21.9	3.11	1.50
64	412	10.0	3.44	1.07	20.1	3.09	1.49
100	607	11.3	4.01	1.26	19.6	3.08	1.48
144	1039	12.5	4.48	1.42	19.3	3.05	1.47
196	1400	13.8	4.89		19.1	3.03	1.46
256	1745	15.0	5.25		18.7	3.02	1.46
324	2227	16.2	5.59		18.6	3.00	1.45
400	2666	17.5	6.01		18.4	2.99	1.45

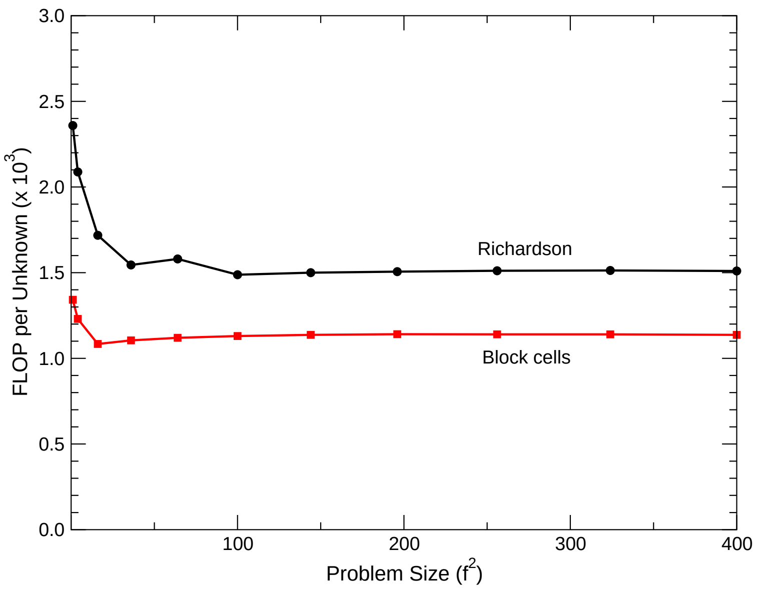
Table 9. Convergence rates of GMRES(20) for the original iron-water problem scalings tabulated as a function of the scaling parameter f.
[a]No preconditioning [b]Richardson [c]Block cells [d]Block lines

even likely, that the dependence on condition number and the slow convergence in problems with thin mesh cells is a reflection of this dependence. If this is true, performance could be improved if multigrid methods are used to precondition the inner CG iterations, although there are additional issues associated with the use of multigrid methods. We will address this issue in future investigations.

6. Summary. We have developed an effective preconditioner for what is essentially a new mixed, discontinuous Galerkin scheme for elliptic operators. The two-level



5(a) Iteration count.



5(b) FLOP count per unknown.

Fig. 5. Number of iterations and optimality results for the iron-water problem with constant scalings as a function of problem size. Two smoothing relaxations are shown.

preconditioner, which is based on a continuous finite element diffusion equation discretization, is robust and effective for a wide range of problems. Using a Fourier analysis to estimate the condition number of a particular, homogeneous problem, we showed that it is possible to predict the relative effectiveness of the preconditioner.

We found that when the mesh contains regions which are thick and diffusive, the preconditioner is most effective and solutions can be computed efficiently. When mesh cells are optically thin or have a high aspect ratio, the relative improvement in convergence rate diminishes. The smoothers based on block Jacobi relaxations help to improve the performance of the preconditioner in that case. Unfortunately, the high setup costs associated with the block line Jacobi smoothers made that it computationally inefficient even though it was the most effective. We can conclude that for future applications a block cell Jacobi relaxation smoother should be used. Still, the method did not scale optimally when the mesh cells become thin. One way to address this is issue might be to precondition the inner iterative CG solution with an optimal method, like multigrid, for example. Perhaps a fixed number of multigrid cycles alone, without CG, might even be better.

Extension to three dimensional unstructured meshes is ongoing. It appears that the most efficient implementation should be a continuous diffusion preconditioner together with a block cell Jacobi relaxation smoother. This should also be efficient if used in a parallel implementation.

Finally, we note that the stability (in the LBB or *inf-sup* sense [5]), as well as the accuracy, of our DFE discretization of the P_1 equations is being investigated.

7. Acknowledgments. We would like to thank Michele Benzi of Emory University for his assistance in matters relating to linear algebra and for the use of his code SPARSELAB. We also would like to acknowledge the use of the SPARSKIT iterative solvers code. Finally, we would like to thank the editors and reviewers of this paper for their comments and suggestions, which we feel helped to greatly improve the quality of this paper.

References.

[1] M. L. Adams and W. R. Martin, *Diffusion synthetic acceleration of discontinuous finite element transport iterations*, Nucl. Sci. Engr., 111 (1992), pp. 145–167.

[2] R. E. Alcouffe, *Diffusion synthetic acceleration methods for diamond-differenced discrete-ordinates equations*, Nucl. Sci. Engr., 64 (1977), pp. 344–355.

[3] R. E. Alcouffe, E. W. Larsen, W. F. Miller, and B. R. Wienke, *Computational efficiency of numerical methods for the multigroup, discrete ordinates neutron transport equations: The slab geometry case*, Nucl. Sci. Engr., 71 (1979), pp. 111–127.

[4] D. N. Arnold, F. Brezzi, B. Cockburn, and D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM J. Num. Anal., 39 (2001), pp. 1749–1779.

[5] K.-J. Bathe, *Finite Element Procedures*, Prentice-Hall, New Jersey, 1996.

[6] C. E. Baumann and J. T. Oden, *A discontinuous* hp *finite element method for convection-diffusion problems*, Comp. Meth. Appl. Mech. Engr., 175 (1999), pp. 311–341.

[7] G. I. Bell and S. Glasstone, *Nuclear Reactor Theory*, Van Nostrand Reinhold Co., New York, 1970.

[8] M. Benzi, C. D. Meyer, and M. Tuma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.

[9] A. Bouras, V. Frayssé, and L. Giraud, *A relaxation strategy for inner-outer linear solvers in domain decomposition methods*, Tech. Report CERFACS TR/PA/00/17, European Centre for Research and Advanced Training in Scientific Computation, 2000.

[10] J. H. Bramble and J. E. Pasciak, *A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems*, Math. Comp., 50 (1988), pp. 1–17.

[11] W. L. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, 1987.

[12] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, eds., *Discontinuous Galerkin Methods*, Springer-Verlag, Berlin-Heidelberg, 2000.

[13] B. Cockburn and C.-W. Shu, *The local discontinuous Galerkin finite element method for convection-diffusion systems*, SIAM J. Num. Anal., 35 (1998), pp. 2440–2463.

[14] B. Davison, *Neutron Transport Theory*, Clarendon Press, Oxford, England, 1957.

[15] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Iterative methods for problems in computational fluid dynamics*, in Iterative Methods in Scientific Computing, R. Chan, et al., Eds., Springer-Verlag, Singapore, 1997, pp. 271–327.

[16] G. H. Golub and Q. Ye, *Inexact preconditioned conjugate gradient method with inner-outer iteration*, SIAM J. Sci. Comput., 21 (1999), pp. 1305–1320.

[17] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[18] C. Hirsch, *Numerical Computation of Internal and External Flows*, vol. 1 *Fundamentals of Numerical Discretization*, Wiley, New York, 1988.

[19] H. Khalil, *Effectiveness of a consistently formulated diffusion-synthetic acceleration differencing approach*, Nucl. Sci. Engr., 98 (1988), pp. 226–243.

[20] E. W. Larsen, *Unconditionally stable diffusion-synthetic acceleration methods for slab geometry discrete ordinates equations. Part I: Theory*, Nucl. Sci. Engr., 82 (1982), pp. 47–63.

[21] ——, *The asymptotic diffusion limit of discretized transport problems*, Nucl. Sci. Engr., 112 (1992), pp. 336–346.

[22] E. W. Larsen and W. F. Miller, *Convergence rates of spatial difference equations for the discrete-ordinates neutron transport equations in slab geometry*, Nucl. Sci. Engr., 73 (1980), pp. 76–83.

[23] E. W. Larsen and J. E. Morel, *Asymptotic solutions of numerical transport problems in optically thick, diffusive regimes II*, J. Comp. Phys., 83 (1989), pp. 212–236.

[24] E. W. Larsen, J. E. Morel, and W. F. Miller, Jr., *Asymptotic solutions of numerical transport problems in optically thick, diffusive regimes I*, J. Comp. Phys., 69 (1987), pp. 283–324.

[25] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhauser Verlag, Basel, 1990.

[26] R. B. Lowrie and J. E. Morel, *Discontinuous Galerkin for stiff hyperbolic systems*, Tech. Report LA-UR-99-2097, Los Alamos National Laboratory, 1999.

[27] G. Meurant, *Computer solution of large linear systems*, in Studies in Mathematics and Applications, vol. 28, Elsevier, Amsterdam, 1999.

[28] J. E. Morel, J. E. Dendy, and T. A. Wareing, *Diffusion-accelerated solution of the two-dimensional S_n equations with bilinear-discontinuous differencing*, Nucl. Sci. Engr., 115 (1993), pp. 304–319.

[29] J. E. Morel, T. A. Wareing, and K. Smith, *A linear-discontinuous spatial differencing scheme for S_n radiative transfer calculations*, J. Comp. Phys., 128 (1996), pp. 445–462.

[30] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Mat. Anal. Appl., 13 (1992), pp. 778–795.

[31] G. C. Pomraning, *A generalized* P-N *approximation for neutron transport problems*, Nukleonik, 6 (1964), pp. 348–356.

[32] ———, *The Equations of Radiation Hydrodynamics*, Pergamon, Oxford, 1973.

[33] A. Ramage and A. J. Wathen, *Iterative solution techniques for the Stokes and Navier-Stokes equations*, International Journal for Numerical Methods in Fluids, 19 (1994), pp. 67–83.

[34] T. Rusten and R. Winther, *A preconditioned iterative method for saddlepoint problems*, SIAM J. Mat. Anal. Appl., 13 (1992), pp. 887–904.

[35] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.

[36] B. Su, *More on boundary conditions for differential approximations*, J. Quant. Spect. Rad. Trans., 64 (2000), pp. 409–419.

[37] T. A. Wareing, J. M. McGhee, J. E. Morel, and S. D. Pautz, *Discontinuous finite element S_n methods on three-dimensional unstructured grids*, Nucl. Sci. Engr., 138 (2001), pp. 256–268.